Learning the Discount Factor in Inverse Reinforcement Learning with Application to Animal Behaviour

by

Litong Zheng

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Learning the Discount Factor in Inverse Reinforcement Learning with Application to Animal Behaviour

Litong Zheng
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2023

Abstract

Inverse reinforcement learning (IRL) focuses on recovering the agent's objective function through observed demonstrations, enabling us to better understand human and non-human animals' decision-making processes. However, an important aspect of decision-making—how we discount the future reward—has not been well-studied in IRL.

In this thesis, we study the problem of IRL with unknown discount factor and develop its first application in animal behaviour. We investigate an existing work on IRL with unknown discount factor in detail, correcting several minor errors in the formulation and examining different mathematical characteristics of the problem. We propose an alternative algorithm for the correct gradient calculation, examine the convexity of the objective function in the optimization problem, and discuss the connections among different formulations of the objective functions seen in the literature. We then apply the framework to studying wild vervet monkeys' foraging behaviour in two different scenarios: foraging alone or in competition. We formulate and solve each scenario as an IRL problem. Our experimental results provide novel insights into vervet monkeys' cognitive decision-making process. We present several suggestions on future foraging experiment design to improve the current mathematical models.

Dedicated to my family.

Acknowledgements

I would like to thank my supervisor Prof. Christopher Beck for his invaluable advice and unwavering support. Thank you, Prof. Beck, for your constant encouragement to strive for excellence.

A special thank you goes to Tan, who has been an exceptional mentor. Your insights and feedback have greatly improved this thesis.

Thank you to our collaborators, Julie and Jean, for the interesting discussions on animal behaviour and for providing the data used in this thesis.

Thank you to Aaron for your thoughtful feedback on the pre-procedure fasting project. Your insights have played a pivotal role in shaping this work.

I would like to thank my committee members, Prof. Chi-Guhn Lee and Prof. Elias Khalil, for their valuable comments during my defense.

Thank you to everyone at TIDEL for all the discussions about research. I wish you all the best in your future endeavours.

I would also like to thank my friends, Christina, Kailyn, and Sheree, for their support and encouragement.

Finally, I would like to express my gratitude to my family, without whom this would not have been possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Decision-making is an intrinsic part of human and non-human animals' lives. As humans, we face decisions such as determining the best route from home to work, while animals may have to choose between pursuing a particular food source or avoiding a potential predator. These decisions reflect our goals and preferences. Consider a modified version of the Stanford marshmallow experiment [77], which originally studied delayed gratification in children. In this scenario, children first need to choose their preferred reward, either a marshmallow or a pretzel stick. Then, they can decide whether to have this reward immediately or, if they can resist the temptation and wait for 15 minutes, receive two marshmallows or pretzel sticks. These choices reflect two fundamental aspects of decision-making: our preferences for the reward itself (e.g., sweet or salty) and how we discount the value of future rewards against immediate gratification (eat now or wait for a greater reward).

In machine learning, the reinforcement learning (RL) paradigm simulates the decision-making process by training an agent to take actions within an environment to maximize its cumulative reward [103]. RL problems are typically formulated as finite Markov decision processes (MDPs) (formally defined in Section 2.2) that discretize the environment into states and an agent's actions in a state result in it receiving an immediate reward and triggering a state transition. For example, we can model our modified marshmallow experiment as an MDP in which the agent is the child and the actions are choosing a marshmallow or a pretzel stick, and to eat now or to wait.

While RL has proven to be a powerful approach in training autonomous agents and developing intelligent systems [57], one of the major challenges in RL is the manual specification of rewards. Designing a reward function that accurately captures all relevant aspects of the incentive often requires expert domain knowledge and fine-tuning [46]. This limitation becomes particularly pronounced in applications with high complexity, such as autonomous driving, where numerous factors need to be considered to ensure safe and efficient navigation [97, 117]. However, since our goal is to train an agent that drives like a human, a more promising approach may be to extract the reward function from demonstrations of human drivers directly. This field of research, called inverse reinforcement learning (IRL) [92], aims to infer the agent's preferences through observed behaviour. For example, if a child participated in our experiment ten times, assuming each trial is independent, they may choose the marshmallow and pretzel stick five times each and choose to have the treat immediately every time. IRL seeks to learn the preferences and the weights of the preferences that lead to such observed behaviour. Compared to RL, IRL does not require the manual specification of the agent's

preferences.

Over the past two decades, IRL has gained significant attention from researchers and has been applied to fields such as robotics [4, 97, 63], human behaviour [71, 120, 31], and animal behaviour [118, 48, 13]. However, most IRL studies focus solely on learning the reward preferences and overlook the degree to which the agent discounts future rewards. That is, IRL techniques are often applied to recover whether the children prefer marshmallows or pretzel sticks, while assuming a particular level of discounting for future rewards. Intelligent decision makers, both humans and non-human animals, value the future differently from one another. To our knowledge, there is only one work (a paper by Giwa and Lee [40] and the accompanying PhD thesis by Giwa [41]) that seeks to recover the reward weights and the discount factor simultaneously.

This thesis focuses on recovering the decision maker's reward preference and discount factor jointly through demonstrations and develops the first application of this problem in animal behaviour using IRL. We first examine the work of Giwa and Lee [40, 41] in detail. Then, we build upon and apply this framework to study the foraging behaviour of wild vervet monkeys. We first cast the foraging experiment as an IRL problem, identifying factors that affect the monkey's choices (the type of food it gets and how far it has to travel to get the food). We then apply our IRL framework to solve this problem. The recovered parameters provide interpretable and novel insights into wild vervet monkeys' cognitive decision-making process.

This thesis confirms that the weight parameters and discount factor can be recovered jointly in an IRL problem and corrects errors in the previous formulation. We also show that IRL is a viable tool for animal behaviour studies and we hope that it can serve as a starting point for more interdisciplinary research between the field of animal behaviour and the IRL community.

1.1 Organization of Thesis

Chapter 2 provides background necessary to understand the thesis. First, the chapter presents the formal definitions of finite MDPs and IRL, and notation that will be used in this thesis. Then, the chapter reviews the IRL literature, covering successful applications and algorithms in both single-agent and multi-agent problems. The chapter closes with a discussion on the limitations and challenges in IRL.

Chapter 3 examines and analyzes the current method of solving IRL with unknown discount factor problems [40, 41] in detail. This chapter introduces the Maximum Entropy IRL framework by Ziebart et al. [129], the foundation of Giwa and Lee's work [40]. We provide a complete derivation of the original MaxEnt IRL optimization problem and correct several minor errors in Giwa and Lee's formulation. In particular, we propose a time-indexed algorithm for computing the gradients required to solve the optimization problem. Next, the chapter investigates a claim about the convexity of the objective function and demonstrates that it is not trivial to prove this claim mathematically. Finally, the chapter closes with a discussion on different objective function formulations in MaxEnt IRL that we found are often overlooked in the literature.

Chapter 4 studies the wild vervet monkey behaviour when foraging alone. The chapter introduces the original foraging experiment by Arseneau-Robar et al. [10] and describes key characteristics and assumptions we made for this foraging problem. We then model the foraging problem as an MDP and solve the single-agent IRL problem to recover the unknown parameters. Experimental results

show that our model produces behaviour that aligns with observed data from the real-life foraging experiment. More importantly, we show that, when foraging alone, monkeys are farsighted decision makers who only value the preferred food choice. The chapter then provides an extensive discussion on the challenges of applying IRL to this application and alternative modelling options for the foraging problem.

Chapter 5 extends the application in Chapter 4 and studies the monkeys' behaviour when foraging in competition. We treat both monkeys in the trial as the decision makers, modelling the problem in a multi-agent IRL setting. We propose two approaches for modelling a monkey's foraging behaviour as an MDP by embedding the other's actions into the environment. Experimental results show that while the monkeys value the immediate reward similarly to the case of foraging alone, depending on their social status, they value the future differently and exhibit different behaviour. We then discuss the limitations of our approaches and alternatives we can explore for the case of foraging in competition.

Chapter 6 concludes this thesis and presents potential next steps for future research.

1.2 Summary of Contributions

The main contributions of this thesis are as follows:

- We correct several minor errors in the original formulation for MaxEnt IRL with unknown discount by Giwa and Lee [40, 41]. We propose an alternative algorithm for the correct gradient calculation.
- We investigate the convexity of the objective function in the optimization problem. We empirically show that the objective function is neither concave nor convex.
- We demonstrate the relationship between different formulations of the objective functions seen in the literature, deriving conditions for when they are equivalent.
- We develop the first application for studying monkey behaviour when foraging alone as a single-agent IRL problem. Our experimental results align with the behaviour observed in the field experiments and provide novel insights into monkeys' cognitive decision-making process.
- We extend the foraging-alone case and present the first application for studying monkey behaviour when in competition as a multi-agent IRL problem. We propose two methods for extracting and embedding the competitor's actions into the environment. Our experimental results show that monkeys exhibit different behaviour depending on their social status.
- We provide several suggestions for future foraging experiment design and alternative modelling options that may lead to more realistic mathematical models and further collaborations between the fields of IRL and animal behavioural science.

Chapter 2

Literature Review

2.1 Introduction

Reinforcement learning (RL) is a machine learning approach where an autonomous agent learns to achieve a goal through interactions with the environment [103]. The objective of the agent is to maximize the accumulated the reward it receives from the environment in the long run. Although RL has seen significant progress in recent decades, a major drawback is the need for manual specification of the reward function [9]. Often, the reward functions are carefully designed by a human expert based on domain specific knowledge and require fine-tuning [46]. In contrast, inverse reinforcement learning (IRL) refers to a class of problems that aim to recover the reward function an agent is optimizing through observed behaviours [92].

IRL, initially proposed in 1998 [92], falls under the paradigm of Learning from Demonstrations (LfD) in robotics [93, 8] and can be considered as a way to perform imitation learning [52]. LfD and imitation learning focus on recovering the policy or mimicking the demonstrated behaviours, which may not necessarily involve recovering the reward function. IRL is also closely related to the well established concept of inverse optimal control (IOC) in control theory. Ab Azar et al. [1] provide a historical and comparative review on IOC and IRL and defined IRL as "modern IOC". Since LfD, imitation learning, and IOC are not the focus of this thesis, the reader is referred to existing surveys [8, 52, 1] for more details on these topics.

Over the past two decades, IRL has gained significant attention from researchers in different fields. IRL is advantageous as it avoids the process of manually designing the agent's preferences and thus enabling better generalization as a learned reward function from one agent can be transferred to another in a different environment, assuming both agents share the same goal. IRL has been applied to fields including robotic systems [4, 97, 63, 34, 84], human behaviour [71, 120, 31, 54], and animal behaviour [118, 48, 13, 85].

This chapter is organized as follows. Section 2.2 introduces the formal definitions and notations of Markov decision process (MDP) and IRL which are used throughout this thesis. Then, Section 2.3 provides an overview of applications of IRL. Sections 2.4 and 2.5 review IRL algorithms for single-agent and multi-agent problems, respectively. Finally, we conclude this chapter in Section 2.6 by briefly discussing some limitations and challenges in IRL.

2.2 Formal Definitions

2.2.1 Markov Decision Process and Reinforcement Learning

A finite Markov decision process (MDP) [50] is defined as a tuple $\langle S, A, P_{sa}, \gamma, R \rangle$ where

- S is a finite set of N **states**.
- $A = \{a_1, ..., a_k\}$ is a set of k **actions**.
- $P_{sa}(\cdot) : S \to [0, 1]$ are the state **transition probabilities** upon taking action a in state s.
- $\gamma \in [0, 1]$ is the **discount factor**, which quantifies how much the agent values future reward relative to the present.
- $R : S \mapsto \mathbf{R}$ is the **reward function**.

Following the book by Sutton and Barto [103], we focus on RL (and IRL) problems that can be formulated as finite MDPs, namely, problems that satisfy the Markov property. A stochastic process is Markovian if it is memoryless, meaning the future only depends on the present state but not the past states. That is, at time step t, the next state, s_{t+1}, only depends on the state s_t and the action a_t:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = P(s_{t+1}|s_t, a_t). \tag{2.1}$$

The RL problem can be broadly defined as a decision-maker learning to achieve a goal through interactions. The decision maker is called the *agent*, and it interacts with the *environment*. The interactions between the agent and the environment are discretized into distinct time steps, where a state s_t provides a representation of the environment and the agent selects an action a_t, which moves the agent into a new state s_{t+1} while it receives a *reward* r_{t+1}. Figure 2.1 from Sutton and Barto [103] illustrates the interactions.

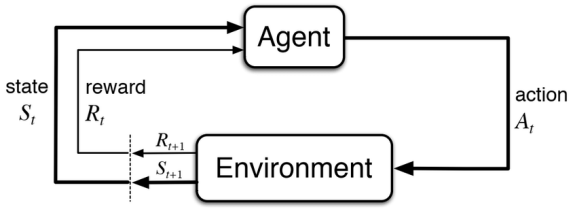


Figure 2.1: General reinforcement learning framework [103].

A *policy* π is a mapping from state to action and $\pi(a|s)$ denotes the probability of taking action a in state s. The value function of an agent's policy V^π represents the value of a given state s as the expected cumulative reward at s following the state sequence $(s, s_1, s_2, ...)$ produced by π, namely

$$V^\pi(s) = E[R(s) + \gamma R(s_1) + \gamma^2 R(s_2) + ...|\pi]. \tag{2.2}$$

We can also write the value function as the Bellman equation [18]:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s')V^{\pi}(s'), \tag{2.3}$$

where $\pi(s)$ represents the best action to take at state s according to a deterministic π. Similarly, the state-action function, or Q-function, represents the value of taking action a at state s and following the policy π:

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s')V^{\pi}(s'). \tag{2.4}$$

The optimal Q-function is defined as $Q^*(s,a) = \sup_{\pi} Q^{\pi}(s,a)$. The goal of solving an MDP is to find an optimal policy π^* such that the value function at each state is simultaneously maximized:

$$V^{\pi^*}(s) = \sup_{\pi} V^{\pi}(s) = \sup_{a} Q^*(s,a), \quad \forall s \in S \tag{2.5}$$

2.2.1.1 Value Iteration and Soft Value Iteration

When the state and action space are small and the environment dynamics are known, one way to solve for the optimal policy is to use value iteration (VI) [103] as shown in Algorithm 1.

Algorithm 1 Value Iteration Algorithm

1: Initialize V arbitrarily
2: **repeat**
3: **for** each $s \in S$ **do**:
4: $v \leftarrow V(s)$
5: $V(s) \leftarrow \arg\max_a \sum_{s'} P_{sa}(s')\big[R(s) + \gamma V(s')\big]$
6: $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
7: **end for**
8: **until** $\Delta < \theta$ (a small positive threshold)
9: Output policy $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s')\big[R(s) + \gamma V(s')\big]$

We first randomly initialize the value function (line 1). Using dynamic programming, we repeatedly calculate the value of each state (line 3–6) until convergence (line 8). Finally, we obtain a policy π based on the value function V by selecting the action at each state that returns the highest value (line 9).

Note that in line 5, we update the $V(s)$ by choosing the action that returns the highest expected return and again in line 9 we pick the best action for each state as the policy. We often choose the max operator in the VI algorithm since it is a non-expansion. A non-expansive operator brings the distance between two points closer or no further apart, and thus the max operator guarantees the convergence of values to a unique fixed point [11]. However, it is important to note that other operators such as the mean operator can also be used. Here, we introduce the softmax operator and the resulting VI algorithm called Soft VI [129], which is relevant later in this thesis.

Analogous to Eq. 2.3 and 2.4, we define the following soft value and Q functions and the policy based on these two functions at state s.

$$V^{Soft}(s) = \sigma \log \sum_{a \in A} e^{\frac{Q^{Soft}(s,a)}{\sigma}} \tag{2.6}$$

$$Q^{Soft}(s,a) = R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^{Soft}(s') \tag{2.7}$$

$$\pi(a|s) = e^{\frac{Q^{Soft}(s,a) - V^{Soft}(s)}{\sigma}} \tag{2.8}$$

The σ in the equations above represents a non-negative value defined as the "temperature", which controls the stochasticity of the policy: a lower temperature leads to more deterministic behaviour while a higher temperature leads to more stochastic behaviour. When $\sigma \to 0$, the optimal policy π^* converges to the deterministic policy from Algorithm 1. Soft VI is presented in Algorithm 2 where we simply replace the max operator in Algorithm 1 with softmax in line 5 and change how we evaluate the action distribution in line 9.

Algorithm 2 Soft Value Iteration Algorithm

1: Initialize V arbitrarily
2: **repeat**
3: **for** each $s \in S$ **do**:
4: $v \leftarrow V(s)$
5: $V(s) \leftarrow \text{softmax}_a \sum_{s'} P_{sa}(s') \big[R(s) + \gamma V(s')\big]$
6: $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
7: **end for**
8: **until** $\Delta < \theta$
9: Output policy $\pi(a|s) = e^{\frac{Q^{Soft}(s,a) - V^{Soft}(s)}{\sigma}}$

2.2.2 Markov Games

The RL framework can also be extended to the case where there are multiple agents, defining a multi-agent RL (MARL) problem. In the multi-agent setting, the agents interact not only with the environment but also with each other, and they can be modelled to act either cooperatively or competitively. MARL problems that satisfy the Markov property can be formulated as Markov games (MGs) [69].

A Markov game (MG) is an extension of an MDP where there is more than one decision maker. Similar to an MDP, a finite MG is a tuple $\langle \mathcal{P}, S, A, P_{sa}, R, \gamma \rangle$, where $\mathcal{P}$ is the set of agents in the game. Without loss of generality, let us assume that there are two players in the game and $\mathcal{P} = 1, 2$. $S = S^1 \times S^2$ and $A = A^1 \times A^2$ are the joint state and action spaces of the two agents. The reward $R = (R^1, R^2)$ where $R^i(s, a^i, a^{-i})$ represents the immediate reward of agent i in state s based on its own and the other player's action ($-i = \mathcal{P} \backslash \{i\}$). The transition probability is similar to an MDP $P_{sa} : S \times A \to S$; and finally $\gamma = (\gamma^1, \gamma^2)$ is the discount factor for each agent.

To solve a Markov game, we can either consider the other agent $-i$ as part of the environment (i.e., embedded in the transition probabilities) or we can model all agents as decision makers in the game. In this thesis, we choose the former method, and so the game reduces to a single-agent MDP and can be solved using VI or other algorithms. However, if we choose to model all agents, we denote π^i as the policy of player i, and the optimal policy $\pi^{*,i}$ maximizes agent i's expected sum of

discounted rewards with respect to its opponent $-i$'s policy. Namely,

$$\pi^{*,i} = \text{argmax}_{\pi^i} V^{\pi^i}(s), \quad \forall s \in S \tag{2.9}$$

where

$$V^{\pi^i}(s) = E_{\pi^{-i}}[R(s, a^i, a^{-i}) + \gamma^i V^{\pi^i}(s')]. \tag{2.10}$$

The game is in Markov perfect equilibrium when each agent's policy is optimal given the policies of the other players. Depending on the type of the game (e.g., general-sum or zero-sum game), different algorithms can be applied to solve the game such as minmax-Q [69] for two-player zero-sum game or Nash-Q for general-sum games [51].

2.2.3 Inverse Reinforcement Learning

In IRL, the goal is to find a reward function that best explains the observed behaviours of an expert, E. Russell [92] defines an IRL problem as:

Given 1) measurements of the expert agent's behaviour over time, 2) measurements of the sensory inputs to the expert; 3) a model of the environment.

Determine the reward function that is being optimized.

More formally, assume the expert interacts with the environment according to its optimal policy π^E, which may or may not be known to the subject agent, whom we call the learner, L. An IRL problem is then defined as follows. Given an MDP without reward, MDP$\backslash R_E$, and a set of M trajectories $\xi = \{\tau_i\}_{i=1}^M$, where $\tau = \{(s_0, a_0), (s_1, a_1), ..., (s_j, a_j)\}$, determine the reward function $\hat{R}_E$.

Further, we define $\boldsymbol{\phi}(s)$ as the feature vector of state s, and we rewrite the reward function (if linear) as the weighted sum of features:

$$R(s) = w_1\phi_1(s) + w_2\phi_2(s) + ... + w_k\phi_k(s) = \mathbf{w}^\top \boldsymbol{\phi}(s)$$

where ϕ_i and w_i represent the i^{th} feature and its associated weight, respectively, and so the IRL problem becomes one of finding the weight vector $\mathbf{w}$.

Like many inverse problems, IRL is ill-posed as there is an infinite number of reward functions that can explain the observed behaviours. One trivial example is when the reward is 0. We discuss different IRL algorithms and how they address the ill-posedness through different techniques such as regularization in Sections 2.4 and 2.5.

2.3 Applications of Inverse Reinforcement Learning

One of the greatest appeals of IRL is that it enables agents to learn from data directly without the researchers having to explicitly specify a reward function. Thus, IRL has attracted significant attention in the past decades from different research communities. In this section, we provide an overview of IRL applications, focusing on three areas: robotics, human behaviour, and animal behaviour.

2.3.1 Robotics

One of the earliest and most notable applications of IRL was to train an autonomous agent to fly a toy helicopter by learning from demonstrations of a human expert [4]. A reward function with 24 features, which produces a policy that is close to the expert's behaviours, is recovered and the authors demonstrate that simple helicopter manoeuvres such as flip and roll can be learned in just a few iterations. Subsequently, the same research group presented the learning of more challenging aerobatic manoeuvres [2, 3, 29]. An autorotation descent and landing, which is required when the helicopter's engine fails [2], as well as one of the most challenging helicopter manoeuvres called "chaos" [3], can be learned via IRL.

Another application that fits well into the IRL framework is autonomous driving. The goal of this task is to train an agent to drive like a human and there are many components one needs to consider when driving. Thus, it is difficult to manually design a reward function that would satisfy all the requirements and it is more promising to apply autonomous learning instead. Sharifzadeh et al. [97] apply IRL with deep Q-network (DQN) to a highway driving problem and show that the algorithm achieves satisfactory results after a few iterations. Kuderer et al. [63] use IRL to learn driving styles from demonstrations using features that represent real drivers' preferences such as desired speed and acceleration. Similarly, Wulfmeier et al. [116] propose an IRL algorithm that uses deep learning to recover the reward function and apply the algorithm to learn a large-scale cost function to test its real-world applicability in the context of autonomous driving [117]. Wu et al. [115] apply sampling-based IRL to recover the reward function using real world traffic data in a continuous domain and test the algorithm on two different driving scenarios—non-interactive and interactive. The authors show that using scenario-specific features, the proposed algorithm produces more accurate predictions compared to the baseline IRL algorithm. More specific driving style learning such as car-following preferences [125], lane change preferences [70], and behaviours at intersections [114] have also been studied.

As mobile robots become more common in the human environment, creating robots that are socially compliant is another research area that has gained popularity in the IRL community [34, 83]. Robot navigation often focuses on tasks in crowded environment and uses trajectories demonstrated by humans (either by using real human behaviours or controlling the robots) such that robots learn to obey certain social norms that people follow. In some applications in this field, IRL plays an indirect role in training the robots. For example, Ziebart et al. [130] use IRL to provide predictions of a pedestrian's movement, which are then incorporated into a hinderance-sensitive robot's planner, so that the robot can plan its path to avoid the pedestrian. Similarly, Fahad et al. [34] use IRL to learn how pedestrians navigate with the goal of extending the learned reward function to a robot performing a navigation task in crowded spaces. Other than learning to predict pedestrian trajectories and to avoid collision, Okal and Arras focus on learning socially normative navigation behaviours, such as learning to stop if there are people engaging in a photography activity ahead of the robot [83]. Different from the previous applications, Okal and Arras apply IRL directly to the planning task, where demonstrations are robot behaviour collected in different scenarios [83]. Vasquez et al. [110] present a systematic and experimental comparison of different features and IRL algorithms used for the robot navigation task, and evaluate the impact on the learning outcomes. Other navigation learning tasks that do not involve interactions with human include path planning for planetary rovers [84] and for a quadruped robot [127].

In summary, IRL has proven to be a valuable framework in various robotic applications. From learning to fly a toy helicopter to learning to drive and robot path planning, real-world tasks are often too complicated for humans to manually specify a reward function that produces satisfactory results. However, demonstrations from human experts for these tasks are abundantly available (e.g., real-life driving data), making IRL a useful tool since it can extract insight automatically from given data in the form of the reward in the environment.

2.3.2 Human and Animal Behaviours

In the previous section, we mentioned the use of IRL on learning pedestrian behaviours for social robot planning. In this section, we review a wider range of applications of IRL on learning human and animal behaviour, where the goal of recovering the reward function not only provides a basis for predictions but also potentially helps us better understand the humans' and animals' decision-making process.

2.3.2.1 Human Behaviours

One of the earliest applications using human data is perhaps by Ziebart et al. [129], where the authors apply IRL to predict taxi drivers' route choices in a large road network. Similarly, Liu et al. [71] study driver behaviour in more specific scenarios, such as when they have no passengers. Muelling et al. [78] identify important elements of a table tennis player's strategy by learning from demonstrations from skilled and naive human players. Rhinehart et al. [90] develop an algorithm that forecasts behaviours using data collected by a wearable camera. The algorithm, called "Discovering Agent Rewards for K-futures Online" (DARKO), is an online version of IRL as it continuously learns to predict human behaviours from a stream of data input. DARKO not only predicts a person's physical trajectory but also their semantic goals. Das and Lavoie [31] study how feedback that a user receives on social media platforms affects their behaviours as a form of IRL. The authors model the feedback (e.g., number of replies a user receives) as features and learn the coefficients in the reward function which explain the user's behaviour. Yang et al. [120] apply IRL to predict human's attention during visual search. The algorithm learns the reward function through sequences of gaze fixations on an image, and the authors show that IRL has the closest performance to humans compared to heuristic and random methods tested in the study. Tastan and Sukthankar [105] employ IRL in the context of creating a bot for first-person shooter games. Learning from demonstrations provided by human experts, the authors report that their bot exhibits the most "human-like" behaviour compared to other baselines.

Besides learning extrinsic factors that guide human behaviours (e.g., feedback on social media from Das and Lavoie [31]), researchers have also used IRL to study the intrinsic factors from a cognitive science perspective. Jara-Ettinger [54] draws a loose connection between Theory of Mind (ToM) [32] and IRL. ToM refers to humans' abilities to reason about other people's mental state based on how they behave, similar to how IRL models are trained on demonstrations to learn a reward function that explains the observed behaviour. The author discusses how our beliefs and desires can be modelled as the environment and the reward function, which together determine what we intend to do or what the policy should be. Baker et al. [16] pose action understanding [39] as an "inverse planning" problem. The same group of researchers later propose the concept

of Bayesian ToM (BToM) [15, 17], which adopts a Bayesian approach to infer the observed agent's goals. Moreover, Wang et al. [111] apply IRL to learn a reward function that incorporates different motivations that drive various behaviours during gameplay. The study shows that, other than trying to win the game, players are also motivated by factors such as socializing with other players [111].

To summarize, IRL has been applied to study both extrinsic and intrinsic factors that drive human behaviour. For extrinsic motivations, applications cover diverse fields such as driving behaviour, sports, and social media, while for intrinsic motivations, researchers have focused more on mapping concepts in cognitive science such as goals and intents to the IRL framework.

2.3.2.2 Animal Behaviours

Applications of IRL in behavioural ecology and animal behaviour have received comparatively less attention than studies on human behaviour, potentially due to difficulty in obtaining behavioural data.

Yamaguchi et al. [118] investigate the behavioural strategies of *Caenorhabditis elegans* (C. elegans) on a thermal gradient. The authors use IRL to obtain the behavioural strategies of different types of worms (e.g., fed worm, starved worms, and thermosensory neuron-deficient worms) and find that fed worms tend to migrate towards the cultivation temperature (i.e., the temperature that the worms were kept in before the experiment), while starved worms tend to migrate away from the temperature. Hernandez-Reyes et al. [48] apply deep IRL [116] to identify male silk moth's strategies for olfactory searches. The experiment was conducted using a virtual reality (VR) device, exposing the silk moths to different stimuli such as wind in a controlled environment. The results show that the IRL algorithm outperforms the baseline methods and have better generalizability. More recently, Ashwood et al. [13] use dynamic inverse reinforcement learning (DIRL) to study the behaviours of mice exploring a labyrinth. In the labyrinth navigation experiment, trajectories of the mice were collected via video [91], capturing the precise positions of the mice and recording the timestamps. Thus, unlike typical IRL algorithms, DIRL recovers a time-varying reward function that characterizes and provides explanation of the animal's behaviours at different time points. Yu et al. [122] and Schafer et al. [94] study collective animal behaviours under the multi-agent IRL framework.

While the studies mentioned above were all conducted in a laboratory setting, there have also been studies on animals in the wild. Pinsler et al. [85] use GPS data to recover the reward function of pigeons. The authors model the birds' behaviours in a flock by treating each bird as a separate MDP. By assuming each bird in the flock is maximizing its own reward function, different sets of learned feature weights are obtained for each bird. The recovered reward functions successfully generate flock-like behaviours, which not only explains the flock's behaviour but can also be potentially applied to swarm robotics, where multiple agents follow similar dynamics to complete tasks. Another bird related application is proposed by Hirakawa et al. [49]. This study aims to address the issue in animal behaviour research where parts of recorded data are missing due to factors such as device malfunctions. Using GPS data collected for streaked shearwater's migration trajectories, the authors recover the seabirds' reward function. The learned reward function is then used to predict the animal's behaviour and therefore fill in the gaps in trajectories.

Though there have not been many IRL studies on animal behaviour, existing literature covers a broad range of animals from worms to seabirds. Similar to applications in human behaviour,

researchers utilize IRL as a tool to better understand and predict animal behaviour.

2.4 Solution Methods for Single-agent IRL

In many IRL algorithms, we often assume that the dynamics of the environment are known to the learner and the features of the reward function are given. The structure of the reward function R_E is predefined, taking on forms such as a linear combination of weighted features [129, 5]. Thus, to learn the parameters in the reward function, many IRL methods require repeatedly solving an MDP using the intermediate learned reward function and updating the parameters after each iteration. A general template of IRL algorithms from a recent survey by Arora and Doshi [9] is shown in Algorithm 3. As we will see in this section, while earlier IRL algorithms address the ill-posedness issue by optimizing a predefined margin between the learner's and the expert's behaviour, algorithms that are more popular now tend to resolve the issue using a probabilistic approach.

Algorithm 3 General IRL process [9]

Input: MDP$\backslash R_E$, and a set of M trajectories $\xi = \{\tau_i\}_{i=1}^{M}$, state feature $\boldsymbol{\phi}$
Output: reward function $\hat{R}_E$

1: Initialize reward function parameters $\mathbf{w_{ini}}$.
2: Solve the MDP with the current reward function and produce an intermediate learned policy π^L.
3: Update the parameters $\mathbf{w}$ to bring the learned behaviour closer to the observed.
4: Repeat Step 2 and 3 till desired convergence criteria is met.

In this section, we review several fundamental algorithms and their variations for solving the single-agent IRL problem.

2.4.1 Margin-based IRL

We refer to the algorithms presented in this section as "margin-based" because they focus on the goal of finding a reward function that either maximizes the gap between observed behaviour better than the next best behaviour or minimizes the difference between the learner and the expert. The earliest algorithm for solving IRL problems is presented by Ng and Russell [82], which takes a given policy π as input and finds a reward function that makes π optimal. The authors first derive a constraint that the reward function must satisfy, namely,

$$(\boldsymbol{P_{a_1}} - \boldsymbol{P_a})(\boldsymbol{I} - \gamma \boldsymbol{P_{a_1}})^{-1}\boldsymbol{R} \succeq 0 \tag{2.11}$$

where $\boldsymbol{P_a}$ is the transition probability matrix of action a (i.e., a $S \times S$ matrix that represents the transition probabilities of every state pair by taking action a), action a_1 represents the policy $\pi(s) \equiv a_1$ and $\boldsymbol{R}$ is the reward vector. Eq. 2.11 is derived from $\boldsymbol{V}^{\pi} = (\boldsymbol{I} - \gamma \boldsymbol{P_{a_1}})^{-1}\boldsymbol{R}$ and so if we rearrange Eq. 2.11, we get

$$\boldsymbol{P_{a_1}}\boldsymbol{V}^{\pi} \succeq \boldsymbol{P_a}\boldsymbol{V}^{\pi}$$

which is the condition of π being optimal—every action following π, a_1, has a higher or equal value than all other actions a for all states $s \in S$. However, the authors point out that this condition leads to many degenerate solutions, including the trivial case where $R = 0$. Thus, the author propose using

a linear program (LP) to find a reward function R that makes π optimal and maximizes the sum of the differences between the value of the best action and the next best action value over all states, namely, $\sum_{s \in S} Q^{\pi}(s, a^*) - \max_{a \in A \setminus a^*} Q^{\pi}(s, a)$ with Eq. 2.11 as a constraint. Further, the authors believe that small rewards are "simpler and therefore more preferable" [82], thus a penalty term is added to the objective function as regularization for the value of the reward function. Extensions to infinite state space and the case when trajectories from a policy π were given instead are also presented in the paper.

Another margin optimization based algorithm takes the difference between the learner's and the expert's expected feature count as the margin and seeks to minimize it. Given that the reward function is formulated as a linear combination of features, apprenticeship learning via IRL by Abbeel and Ng [5] proposes the notion of *feature matching*:

Given the expert's feature expectations, denoted as μ_E.

Find a reward function that makes the performance of the learner's policy, π^L, close to that of the expert's and thus $||\mu(\pi^L) - \mu_E||_2 < \epsilon$.

Two algorithms were presented in the paper, **max-margin** and **projection**, to find the reward function. The max-margin algorithm is very similar to the optimization problem presented in Ng and Russell [82] except that the weight vector is regularized under the L_2 norm, and thus requires a quadratic program (QP) to solve it instead of an LP. The **projection** algorithm simplifies max-margin and does not require a QP solver. Syed and Schapire [104] propose the multiplicative weights for apprenticeship learning (MWAL) algorithm, which extends apprentice learning using a game-theoretic approach. MWAL models the problem as a two-player zero-sum game between the learner and the environment, where the former chooses a policy to maximize its performance while the latter selects a reward function. Neu and Szepesvári [81] propose a gradient method to learn a reward function that minimizes the action distribution difference between learned and expert policy's (i.e., $\pi^L(a|s) - \pi^E(a|s)$). The concept of feature matching later became fundamental in other IRL algorithms as the way to ensure the learner's policy matches the performance of the expert [129, 88, 116].

Finally, another margin-based optimization algorithm is the maximum margin planning (MMP) proposed by Ratliff et al. [88]. Similar to Ng and Russell [82], MMP also seeks a reward function that makes the demonstrated behaviour better than other possible behaviour by a margin. However, MMP does not assume that the demonstrations come from a single policy π but rather focuses on matching the behaviour demonstration by demonstration. The authors define a loss function that penalizes taking actions different from the expert's. The loss function can take on different forms and thus can scale the margin differently.

2.4.2 Bayesian IRL

Different from using margin optimization as a way to resolve the issue of ill-posedness, Bayesian IRL (BIRL) [87] approaches the issue from a probabilistic point of view. BIRL treats the demonstrations from the expert as evidence to update a prior on the reward function. The algorithm assumes that the expert maximizes its total accumulated reward, namely, at each state the expert picks the action that has the highest Q-value, Q^*. Further, BIRL assumes that the expert is following a stationary policy that does not change over time. Given a demonstration with length k, $\tau = \langle (s_1, a_1), ..., (s_k, a_k) \rangle$, we

can make the following assumption given an agent's stationary policy:

$$P(\tau|R) = P((s_1, a_1)|R)P((s_2, a_2)|R)...P((s_k, a_k)|R).$$

Here, we assume that the state-action pairs are independent of one other. Thus, the probability of a trajectory is the product of the probabilities of the state-actions in that trajectory. Moreover, BIRL models $P((s_i, a_i)|R)$ as a Boltzmann distribution based on Q^*, i.e., $P((s_i, a_i)|R) = \frac{1}{Z_i} e^{\alpha Q^*(s_i, a_i, R)}$, where Z_i is the normalization function and α represents the inverse temperature in Boltzmann distribution—the higher α is, the more rational the agent would be. Thus, the likelihood of the agent producing demonstration τ given reward function R is

$$P(\tau|R) = \frac{1}{Z} e^{\alpha \sum_{i=1}^{k} Q^*(s_i, a_i, R)}.$$

Using Bayes' Theorem, we have

$$P(R|\tau) = \frac{P(\tau|R)P(R)}{P(\tau)}.$$

Several distributions for $P(R)$ are suggested in BIRL, such as Gaussian, Laplace, and Beta distribution depending on the specific characteristics of the problem. Computing $P(\tau)$ is difficult, and thus a Markov Chain Monte Carlo (MCMC) algorithm is used to obtain the posterior mean of R. Lopes et al. [72] adopt the BIRL framework and extended it to an active learning setting. Unlike previous algorithms where the provided demonstrations are fixed, the agent is allowed to query the expert for some specific states.

Under the Bayesian framework, Choi and Kim [27] propose using the maximum a posterior (MAP) estimation for the reward function. The authors point out that using the posterior mean reward function may result in an optimal policy inconsistent with the observed data since the objective function integrates over the entire reward space. Moreover, the authors show that many IRL algorithms can be cast as BIRL using MAP and propose a gradient method for finding the MAP reward function. Choi and Kim later extend their work and present hierarchical BIRL (HBIRL), which imposes a prior on the confidence parameter α and a hyper-prior on the prior of the reward function [26].

Michini and How [76] present a modified BIRL algorithm that enables scaling to large spaces. The original BIRL algorithm suffers from the "curse of dimensionality" [76] since it requires re-solving the MDP in every iteration. The improved algorithm suggests that the inference task should focus on states that are in the demonstrations rather than the entire state space. The authors show that their algorithm significantly reduces convergence time while maintaining solution quality comparing to the original BIRL [76]. Michini and How also propose another algorithm called Bayesian Nonparametric IRL (BNIRL) [75]. In BNIRL, the data is partitioned into different "sub-demonstrations" automatically, and we infer a set of simpler reward functions corresponding to each partition. Choi and Kim [28] present a nonparametric Bayesian IRL algorithm for inferring multiple reward functions. While each trajectory is assumed to be generated from one reward function, the authors study the scenario where the demonstration set might consist of trajectories from different reward functions.

Other algorithms related to BIRL or the Bayesian approach in general include robust BIRL proposed by Zheng et al. [126] to handle sparse behaviour noise and Gaussian process IRL (GPIRL)

by Levine et al. which represents the reward function as a nonlinear function of features modelled as a Gaussian process [66]. More recently, Brown and Niekum [23] propose a deep Bayesian IRL algorithm that can scale to more complex problems.

2.4.3 Maximum Entropy IRL

Similar to BIRL, another popular algorithm that attacks ill-posedness from a probabilistic view is Maximum Entropy IRL (MaxEnt IRL) proposed by Ziebart et al. [129]. Entropy, in information theory, is a measure of the average amount of "information" conveyed by a random variable [96]. For a random variable X that takes values in a set $\mathcal{X}$ and is distributed according to $P : \mathcal{X} \to [0, 1]$, the entropy of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log P(x).$$

MaxEnt IRL applies the principle of maximum entropy [55] and produces a distribution of trajectories parameterized by the weights of the features that is the least wrong while satisfying the constraints in the optimization problem:

$$\text{maximize} \quad \sum_{\tau} -P(\tau) \log P(\tau) \tag{2.12}$$

$$\text{subject to} \quad \sum_{\tau} P(\tau) = 1 \tag{2.13}$$

$$\mathrm{E}_{\tau \sim P}[\phi(\tau)] = \mathrm{E}_{\tau \sim \xi}[\phi(\tau)] \tag{2.14}$$

The objective of the optimization problem is to find a distribution of all possible trajectories that maximizes the entropy. The first constraint represents the axiom of probability while the second constraint represents the concept of feature matching [5] defined in Section 2.4.1, where the left-hand side represents the feature expectation of the trajectory distribution, and the right-hand side represents the feature expectation of the given demonstration, which can be empirically obtained through $\mathrm{E}_{\tau \sim \xi}[\phi(\tau)] = \frac{1}{|\xi|} \sum_{\tau \in \xi} \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t)$.

The Lagrangian relaxation of this formulation shows that the distribution of trajectories belongs to the exponential family, and thus maximizing the entropy is equivalent to finding the weights of the reward function that maximizes the likelihood of the given trajectories:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{\tau \in \xi} \log P(\tau; \mathbf{w})$$

where

$$P(\tau; \mathbf{w}) = \frac{e^{\mathbf{w}^\top \phi(\tau)}}{Z}$$

and Z is the partition function. Ziebart et al. [129] solve this optimization problem through gradient descent, where the gradient can be expressed as the difference between the expert's expected feature count and the learner's.

While Ziebart et al. [129] assume the reward function is a linear combination of the features, Wulfmeier et al. [116] generalize MaxEnt IRL to non-linear reward function by using a neural network to approximate the reward function, in an approach called Deep MaxEnt IRL. Deep MaxEnt

IRL uses the same gradient as Ziebart et al. [129] in backpropagation to update the parameters in the neural network.

Audiffren et al. [14] propose an algorithm that combines MaxEnt IRL with semi-supervised learning [109]. The Maximum Entropy Semi-Supervised Inverse Reinforcement Learning (MESSI) algorithm takes not only the trajectories from the expert but also "unsupervised trajectories" that may or may not be generated by the expert. MESSI adds a penalty term $R(w)$ that penalizes weights that produce different rewards for similar trajectories and updates the weight parameter in the similar fashion to MaxEnt IRL [129].

Ziebart et al. [128] extend MaxEnt IRL to maximizing causal entropy (MCE IRL) for stochastic processes. Mai et al. [73] generalize MCE IRL to not only recover the reward function but also partially recover the underlying structure among the states that impacted the expert's decision. Bloem and Bambos extend MCE IRL to the infinite horizon setting [20].

While in the original MaxEnt IRL algorithm, Ziebart et al. proposed a simple dynamic program to estimate the partition function Z, other algorithms have used sample-based techniques such as importance sampling [22, 58, 36]. Boularias et al. propose relative entropy IRL (REIRL) which minimizes the relative entropy (or Kullbach-Leibler (KL) divergence). The objective function of REIRL is $\min \sum_\tau P(\tau) \log \frac{P(\tau)}{Q(\tau)}$ where $Q(\tau)$ is an empirical trajectory distribution obtained by a baseline policy while $P(\tau)$ is the distribution from the learner's policy. Through importance sampling, REIRL can be used to solve model-free problems using stochastic gradient descent [22]. A recent study by Snoswell et al. [100] proposes a new formulation of MaxEnt IRL using KL divergence and therefore unifies MaxEnt IRL and REIRL. Other sample-based algorithms include path integral IRL (PI-IRL) by Aghasadeghi and Bretl [6] and guided cost learning (GCL) proposed by Finn et al. [36], which both generalize MaxEnt IRL to high-dimentional and continuous state space.

Finn et al. [35] also present a connection between generative adversarial networks (GANs) [44] and MaxEnt IRL. The authors show that when the generator's density is known and assuming the real distribution of given data follows the Boltzmann distribution, GAN and sample-based MaxEnt IRL are equivalent. Later, Fu et al. build on this work and propose Adversarial IRL (AIRL) [38]. Fu et al. show that the trajectory-centric approach of Finn et al. performs poorly in practice. While Finn et al. infers the distribution of trajectories, AIRL's discriminator learns to discriminate state-action pairs instead, resulting in a more scalable and robust algorithm.

Other IRL algorithms based on the MaxEnt framework include inverse reinforcement learning from failure (IRLF) by Shiarlis et al. [98], which learns from two sets of demonstrations—successful and failed. On top of the feature matching constraint in MaxEnt IRL, IRLF also has a constraint that maximizes the dissimilarity between the feature expectations of the learned feature and the failed set. Kamalaruban et al. propose an interactive teaching algorithm for IRL where a "teacher" can provide demonstrations to the learner based on the performance of the learner's policy [59]. Ashwood et al. [12] propose dynamic IRL (DIRL) which not only learns a time-varying reward function but also recovers the features of each state.

2.4.4 Other IRL Algorithms

Besides the three most popular frameworks presented above, researchers have also proposed algorithms that approach the IRL problem differently. For example, the Structured Classification-based IRL (SCIRL) algorithm by Klein et al. [61] casts IRL as a multi-class classification problem. In

SCIRL, the demonstration set is treated as a training set, where each state-action pair is a data-label pair. The same researchers also develop an IRL algorithm using Cascaded Supervised Learning (CSI), which breaks the IRL problem into two supervised learning problems to infer the reward function. Uchibe combines deep IRL with logistic regression [108]. Cross-embodiment IRL (XIRL) by Zakka et al. [123] enables the agent to learn from video demonstrations where the same tasks have different embodiments (e.g., taking different actions to complete the same task).

2.4.5 Summary

In this section, we reviewed three major frameworks—margin-based algorithms, the Bayesian approach, and the maximum entropy approach—for solving IRL problems as well as their extensions. The common assumptions in these algorithms are: i) the demonstrations are either optimal or near optimal and are complete, meaning the learner has access to all information in the trajectories; and ii) the expert is optimizing one goal. While many algorithms use a linear combination of weighted feature to impose some structure to the reward function, non-linear functions were used in the form of neural networks [116] or Gaussian process [66]. The algorithms covered in this section align with the focus of this thesis, where we only consider complete demonstrations with one objective, and we have a complete MDP model for our problem. However, there also exist algorithms that handle incomplete demonstrations, incomplete models, or the case when the expert has multiple intentions. The interested reader is referred to a recent survey [9] as a starting point for these techniques.

2.5 Solution Methods for Multi-agent IRL

As we have seen in the previous section, most of the IRL algorithm focus on a single agent interacting with the environment and MA-IRL only started receiving more attention over the past decade. One can cast a multi-agent problem back to single-agent by choosing to focus on one agent at a time and modelling the other agents as part of the environment. For example, Tian et al. [106] propose a bounded rationality [99] two-player Markov game model where each player is modelled separately with the other embedded in the transition dynamics. However, this approach may not be applicable in certain situations where modelling the interactions between agents is important, such as when the agents need to cooperate to achieve a common goal.

The first MA-IRL study is by Natarajan et al. [80] in 2010. This study assumes there exists a centralized controller whose goal is to maximize the individual agents' joint reward. The authors extend the work of Ng and Russell [82] to learn the reward function of an average-reward MDP. Reddy et al. study a decentralized case where each agent maximizes its own reward and modelled the problem as a general sum stochastic game [89]. Reddy et al. also use Ng and Russell's classical IRL algorithm [82] and the authors assumed that the agents' policies are at a Nash equilibrium. Lin et al. [68] propose a Bayesian framework for solving competitive zero-sum games where the prior of the rewards is assumed to be Gaussian, following Qiao and Beling [86]. Lin et al. [67] later expand this approach to five classes of general sum games under different equilibria. Bogert and Doshi [21] extend the MaxEnt IRL framework, called IRL* to study multi-robot interactions under occlusions, meaning some state-action pairs in other agents' trajectories may be hidden from the learner. Since interactions are sparse in Bogert and Doshi, each robot is modelled as a separate MDP except when

they interact, which is then assumed that they are at a Nash equilibrium. Recently, AIRL has also been generalized to a multi-agent setting (MA-AIRL) by Yu et al. [121].

Šošić et al. [101] study IRL in swarm systems where each agent in the swarm is homogenous and only has access to local information (i.e., the agent can only observe a pre-defined neighbourhood near it). The authors show that the problem can be reduced to a single-agent problem due to the homogeneity assumption. Yu et al. [122] follow the framework developed by Šošić and propose a parameter-sharing AIRL (PS-AIRL) algorithm, which also utilizes the homogeneity of agents in the system and assumes agents share the same parameters in the policy network. Hadfield et al. [47] formulate the value alignment problem as cooperative IRL (CIRL), defined as a two-player game of partial information with a human player and a robot player. In CIRL, the human knows the reward function while the robot initially does not. The goal of CIRL is to make the robot learn and maximize the reward function, hence aligning itself to the human's goal. Zhange et al. [124], conversely, study the case when the agents have misaligned goal and develop non-cooperative IRL (N-CIRL). N-CIRL is formulated as a zero-sum Markov game with one-sided information where only one player knows the true reward function. Zhang et al. apply N-CIRL in a cybersecurity setting where the defender must learn the attacker's intent.

Finally, although the MA-IRL problem is often formulated as a game, existing literature seldom makes the connection between MA-IRL and inverse game theory [64]. In game theory literature, Inga et al. [53] and Mehr et al. [74] both extend MaxEnt IRL [129] to solve inverse dynamic games. Cao and Xie [25] propose a game-theorectic IRL (GT-IRL) framework where both the system dynamics and the reward function of the players are parameterized and unknown. Waugh et al. [112] and Bertsimas et al. [19] study the equilibrium estimation problem from given behaviours, which is effectively equivalent to recovering a reward function that makes the observed behaviour optimal in IRL.

In summary, MA-IRL was introduced roughly a decade after the single-agent IRL problem was proposed in 1998 [92]. MA-IRL algorithms are mostly extensions of existing single-agent IRL algorithms such as MaxEnt IRL [129] or BIRL [87]. While the connection between forward multi-agent RL and game theory is well established [69, 119], MA-IRL and inverse game theory are often seen as separate topics by the two research fields and thus there exists an opportunity to unify the two frameworks.

2.6 Conclusion

In this chapter, we presented an overview of IRL by first providing the formal definitions of related concepts, followed by the applications of IRL. We also present some foundational algorithms for single-agent and multi-agent IRL and how these algorithms address the ill-posedness of the IRL problem. Although IRL has gained significant attention over the past two decades, some limitations and challenges IRL faces include:

- Although IRL avoids the issue of specifying the exact reward function as in RL, the majority of IRL algorithms still require us to choose a set of features and to impose some structure in the reward function (e.g., a linear combination) [36], making the algorithms highly sensitive to feature selection [9]. There exists only a few works that discuss feature construction and selection in IRL [110, 65]. Further, if we choose to represent the reward function using a neural

network [116, 36], avoiding specifying the function structure and hand-designed features, we face the issue of losing the ability to interpret the recovered results. Therefore, there is a trade-off between choosing a feature-based function based on domain knowledge that may not correctly represent the true reward and choosing a more expressive function that works well but loses interpretability.

- As we have seen in Algorithm 3, many popular IRL algorithms still require solving the forward problem in the learning process, which may lead to computational and scaling issues.
- Besides the machine learning community, IRL has also been applied to the field of cognitive science and behavioural science. However, it is worth noting that there appears to be a disconnect with between IRL and other research communities even though concepts similar to IRL have been proposed (e.g., inverse dynamic games, ToM). Building stronger connections among these different fields can potentially lead to novel applications and methods for IRL.

Chapter 3

Learning the Discount Factor in IRL

In the literature, standard IRL algorithms focus on learning weights in the reward function solely. For example, recall the MaxEnt IRL framework [129] presented in Chapter 2, where we see that under the principle of maximum entropy, the goal is to maximize the log likelihood of the observed trajectories $\tau \in \xi$:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{\tau \in \xi} \log P(\tau; \mathbf{w}).$$

It seems intuitive that we can extend the maximum likelihood estimation to be with respect to both $\mathbf{w}$ and γ:

$$\underset{\mathbf{w},\gamma}{\operatorname{argmax}} \sum_{\tau \in \xi} \log P(\tau; \mathbf{w}, \gamma) \tag{3.1}$$

However, to the best of our knowledge, there is only one study that attempts to learn the discount factor along with the weight parameters in the reward function [40, 41]. The authors build upon MaxEnt IRL and present a method of learning both parameters simultaneously. However, there are several minor errors in their proposed formulation, that we discuss in Section 3.3.

This chapter is organized as follows. We first address the importance of the discount factor in the agent's behaviours in Section 3.1. Section 3.2 provides a complete formulation of the modified MaxEnt IRL problem. Then, Section 3.3 presents the corrected version of solving the optimization problem using the extended MaxEnt IRL algorithm based on Giwa and Lee [40]. In Section 3.4, we investigate a claim about Eq. 3.1's convexity made in the study. Next, we discuss the connections among different ways of formulating the likelihood function in Section 3.5. Finally, we conclude this chapter in Section 3.6.

3.1 Motivation

The discount factor γ is often seen as a mathematical convenience to ensure convergence for the value function in an infinite horizon setting [40]. However, the discount factor in an MDP also represents how much the agent values the future. A smaller discount factor value indicates that the agent is more myopic, meaning the agent cares more about the immediate reward than the future

reward, while a larger value represents a larger emphasis on future reward.

Consider the example shown in Figure 3.1. We can construct an MDP where each node can be seen as a state and nodes T_1 and T_2 are terminals. Each arc represents a valid action from each state and the values of the arcs represent the immediate reward an agent receives. Table 3.1 shows the values obtained by solving the MDP under two different discount factors, $\gamma_1 = 0.1$ and $\gamma_2 = 0.9$, using regular VI.

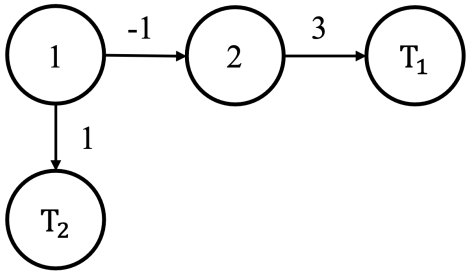


Figure 3.1: A motivating example with four states.

Assume the starting state is node 1, we can see that when γ is small, the optimal policy is to move to node T_2 directly for a reward of 1 (i.e., $\pi(s = 1) = \arg\max_a Q(1, a) = T_2$); when γ is large, the agent will move towards node T_1 despite the negative reward for moving from node 1 to node 2.

Table 3.1: Values of each state-action pair and each state for the example in Figure 3.1.

State	$\gamma_1 = 0.1$	$\gamma_2 = 0.9$
1	$Q(1, 2) = 0.1$	$Q(1, 2) = 1.7$
	$Q(1, T_2) = 1$	$Q(1, T_2) = 1$
	$V(1) = 1$	$V(1) = 1.7$
2	$Q(2, T_1) = 3$	$Q(2, T_1) = 3$
	$V(2) = 3$	$V(2) = 3$

From this simple example, we illustrate how the discount factor affects the agent's decisions. From an IRL perspective, when studying the expert's behaviours, we believe it is reasonable to treat the discount factor as an unknown variable rather than choosing an arbitrary value as most RL and IRL studies. One can imagine that if the expert demonstrations are from a myopic expert, having a large discount factor may result in a reward function that has low interpretability in terms of how each feature affects the agent. As discussed later in this thesis, our application focuses on interpreting real-life animal behaviour, therefore, learning the agent's discount factor allows us to better understand why an agent exhibits certain behaviour. Note that an IRL problem with unknown discount factor is ill-posed: for example, the combination of a 0 reward function and any discount factor $\gamma \in [0, 1]$ is a valid solution to the problem. We adopt a probabilistic approach based on MaxEnt IRL [129] to resolve the ill-posedness issue.

3.2 MaxEnt IRL as Maximum Likelihood Estimation

In this section, we present a complete formulation of the modified MaxEnt IRL problem and how we arrive at Eq. 3.1. The derivations presented in this section are based on Ziebart et al. [129] and

Giwa and Lee [40]. We note that the derivations have been shown in the literature [42], nonetheless, we show the steps required for completeness.

The notations used in this chapter follow the ones presented in Section 2.4.3. Recall ξ is defined as the set of trajectories and $\tau = \{(s_0, a_0), (s_1, a_1), ..., (s_n, a_n)\}$ is a trajectory of length n. $P(\tau)$ is the probability of τ under a policy. Here, we present the optimization problem shown in Chapter 2 again, with the feature matching constraint written explicitly:

$$\max \quad -\sum_{\tau} P(\tau) \log P(\tau) \tag{3.2}$$

$$\text{subject to} \quad \sum_{\tau} P(\tau) \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) = \frac{1}{|\xi|} \sum_{\tau \in \xi} \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) \tag{3.3}$$

$$\sum_{\tau} P(\tau) = 1 \tag{3.4}$$

The optimization problem aims to find a trajectory distribution that maximizes the entropy while ensuring the feature matching requirement between the demonstrations and the learner's policy is satisfied (Constraint 3.3) and the probabilities of all trajectories sum up to 1 (Constraint 3.4). The discount factor is often included implicitly in the expected feature count term. However, there exists formulations in the literature that explicitly include γ. For example, Bloem and Bambos [20] use the expression in Constraint 3.3 when formulating the infinite horizon MCE IRL algorithm, and Gleave and Toyer [42] include γ in the finite horizon MaxEnt IRL formulation to show that having a $\gamma < 1$ would not affect the derivations [42] w.r.t. the weights $\mathbf{w}$. Since γ is one of the unknown variables, we also write γ explicitly in the formulation for clarity.

Next, we present the Lagrangian relaxation of the optimization problem. We first move the two constraints to the objective function with λ and μ as the Lagrangian multipliers:

$$\begin{aligned} \max \quad & -\sum_{\tau} P(\tau) \log P(\tau) \\ & - \lambda^\top \Big(\sum_{\tau} P(\tau) \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) - \frac{1}{|\xi|} \sum_{\tau \in \xi} \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) \Big) \\ & - \mu \Big(\sum_{\tau} P(\tau) - 1 \Big) \end{aligned}$$

Taking the derivative of the objective function with respect to the distribution $P(\tau)$ and setting it to zero, we get

$$\begin{aligned} & - \frac{d}{dP(\tau)} \sum_{\tau} P(\tau) \log P(\tau) \\ & - \frac{d}{dP(\tau)} \lambda^\top \Big(\sum_{\tau} P(\tau) \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) - \frac{1}{|\xi|} \sum_{\tau \in \xi} \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) \Big) \\ & - \frac{d}{dP(\tau)} \mu \Big(\sum_{\tau} P(\tau) - 1 \Big) = 0 \end{aligned} \tag{3.5}$$

We can simplify the first term as

$$\begin{aligned}\frac{d}{dP(\tau)}\Big(\sum_{\tau} P(\tau)\log P(\tau)\Big) &= \sum_{\tau}\Big(\frac{d}{dP(\tau)}P(\tau)\Big)\log P(\tau) + \sum_{\tau} P(\tau)\Big(\frac{d}{dP(\tau)}\log P(\tau)\Big)\\ &= \sum_{\tau}\log P(\tau) + \sum_{\tau} 1\end{aligned}$$

The second term becomes

$$\frac{d}{dP(\tau)}\lambda^{\top}\Big(\sum_{\tau} P(\tau)\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)\Big) = \lambda^{\top}\Big(\sum_{\tau}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)\Big)$$

And finally, the third term

$$\frac{d}{dP(\tau)}\Big(\mu\big(\sum_{\tau} P(\tau) - 1\big)\Big) = \mu\Big(\sum_{\tau} 1\Big)$$

Putting the three derivatives together, we rewrite 3.5 as

$$\begin{aligned}&-\sum_{\tau}\log P(\tau) - \sum_{\tau} 1 - \lambda^{\top}\sum_{\tau}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \mu\sum_{\tau} 1 = 0\\ \Longrightarrow\quad &\sum_{\tau}\Big[\log P(\tau) + \lambda^{\top}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) + \mu + 1\Big] = 0\\ \Longrightarrow\quad &\log P(\tau) + \lambda^{\top}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) + \mu + 1 = 0\\ \Longrightarrow\quad &\log P(\tau) = -\lambda^{\top}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \mu - 1\\ \Longrightarrow\quad &P(\tau) = \exp\big(-\lambda^{\top}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \mu - 1\big)\end{aligned} \tag{3.6}$$

From Eq. 3.6, we can see that $P(\tau)$ is under the exponential family, namely:

$$P(\tau) \propto \exp\big(-\lambda^{\top}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)\big)$$

Since the reward is linear in the features $\phi(s_t)$, we substitute $\mathbf{w} = -\lambda$ back to 3.6,

$$P(\tau) = \exp\left(\mathbf{w}^\top \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t) - \mu - 1\right)$$
$$= \exp\left(\sum_{t=0}^{|\tau|} \gamma^t \mathbf{w}^\top \phi(s_t) - \mu - 1\right)$$

Recall that $\sum_\tau P(\tau) = 1$, and following Giwa and Lee's notations [40], we define the discounted sum of reward as $U(\tau) = \sum_{t=0}^{|\tau|} \gamma^t \mathbf{w}^\top \phi(s_t)$, we have

$$\sum_\tau \exp\left(U(\tau) - \mu - 1\right) = 1$$
$$\exp(\mu + 1) = \frac{1}{\sum_\tau \exp(U(\tau))}$$

Therefore, we arrive at the conclusion that

$$P(\tau) = \frac{e^{U(\tau)}}{\sum_\tau e^{U(\tau)}} = \frac{e^{U(\tau)}}{Z} \tag{3.7}$$

where Z is the partition function. Similar to Ziebart et al. [129], maximizing the entropy of the distribution over all trajectories subject to the feature constraints from demonstrations implies that we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution and parameters $\mathbf{w}$ and γ

$$\mathbf{w}^*, \gamma^* = \arg\max_{\mathbf{w},\gamma} \sum_{\tau \in \xi} \log P(\tau | \mathbf{w}, \gamma). \tag{3.8}$$

3.3 Learning the Weights and the Discount Factor

In Giwa's thesis [41], the optimization problem is solved by using gradient descent directly. The gradients derived in their study have several minor notation errors and, more importantly, one of the algorithms for approximating the partition function Z is incorrect. In this section, we provide the correct gradient derivations and the approximation algorithm for Z.

First, let us derive the gradients of the maximum likelihood with respect to $\mathbf{w}$ and γ (Eq. 3.8). The derivations are not presented in Giwa's thesis [41] and we include the steps here for completeness.

Taking the derivative w.r.t. $\mathbf{w}$, we have:

$$\begin{aligned}
\frac{d}{d\mathbf{w}}\sum_{\tau\in\xi}\log P(\tau|\mathbf{w},\gamma) &= \frac{d}{d\mathbf{w}}\sum_{\tau\in\xi}\log(\frac{e^{U(\tau)}}{Z}) \\
&= \frac{d}{d\mathbf{w}}\sum_{\tau\in\xi}U(\tau) - \sum_{\tau\in\xi}\log Z \\
&= \sum_{\tau\in\xi}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \sum_{\tau\in\xi}\frac{\sum_\tau\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)e^{U(\tau)}}{Z} \\
&= \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \sum_\tau P(\tau)\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)
\end{aligned} \tag{3.9}$$

Similarly,

$$\begin{aligned}
\frac{d}{d\gamma}\sum_{\tau\in\xi}\log P(\tau|\mathbf{w},\gamma) &= \frac{d}{d\gamma}\sum_{\tau\in\xi}\log(\frac{e^{U(\tau)}}{Z}) \\
&= \frac{d}{d\gamma}\sum_{\tau\in\xi}U(\tau) - \sum_{\tau\in\xi}\log Z \\
&= \sum_{\tau\in\xi}\sum_{t=1}^{|\tau|}t\gamma^{t-1}\mathbf{w}^\top\phi(s_t) - \sum_{\tau\in\xi}\frac{\sum_\tau\sum_{t=1}^{|\tau|}t\gamma^{t-1}\mathbf{w}^\top\phi(s_t)e^{U(\tau)}}{Z} \\
&= \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=1}^{|\tau|}t\gamma^{t-1}\mathbf{w}^\top\phi(s_t) - \sum_\tau P(\tau)\sum_{t=1}^{|\tau|}t\gamma^{t-1}\mathbf{w}^\top\phi(s_t)
\end{aligned} \tag{3.10}$$

The last line of Eq. 3.9 is the discounted feature count difference between the expert and the learner. The first term represents the empirical mean of the given trajectories and the second term is the expected discounted feature count from the learner's policy, which generates $P(\tau)$. Eq. 3.10 follows the same structure where the first term can be directly calculated using the demonstrations while the second term comes from the learned policy. Note that Eq. 3.9 is exactly the feature matching constraint we have in the original optimization problem, meaning that the optimal solution of the maximum likelihood problem would be feasible for the original problem. Eq. 3.10 shows a similar matching requirement between the expert and the learner.

The difficulty of computing Eqs. 3.9 and 3.10 lies in obtaining the value of Z, which is required to compute $P(\tau)$. The partition function relies on the knowledge of all feasible paths. However, as the size of the state and action space increases, the number of paths increases exponentially, making the enumeration computationally infeasible. One way to resolve this issue is to compute the expected sum of discounted features over all states (and the discounted sum of discounted reward scaled by time) instead. In Eqs. 3.11 and 3.12, we replace $P(\tau)$ with the expected state visitation frequency, denoted as $P_t(s|\mathbf{w},\gamma)$. The gradients then become:

$$\nabla_\mathbf{w} = \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=0}^{|\tau|}\gamma^t\phi(\tau;t) - \sum_{s\in S}\sum_{t=0}^{T}\gamma^t P_t(s_t|\mathbf{w},\gamma)\phi(s_t) \tag{3.11}$$

$$\nabla_\gamma = \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=0}^{|\tau|} t\gamma^{t-1}\mathbf{w}^\top\phi(\tau;t) - \sum_{s\in S}\sum_{t=0}^{T} t\gamma^{t-1}P_t(s_t|\mathbf{w},\gamma)\mathbf{w}^\top\phi(s_t) \tag{3.12}$$

The gradients shown above follow the same structure as the ones presented in Giwa and Lee [40], where $P_t(s_t|\mathbf{w},\gamma)$ represents the expected state visitation frequency at time step t for the t^{th} state in the trajectory and T is the planning horizon. We can calculate $P_t(s_t|\mathbf{w},\gamma)$ similarly to the algorithm presented in Ziebart et al. [129]. However, since this term is now time-indexed, we need to record the visitation frequency at each time step, instead of summing the state visitation frequency over an arbitrary horizon (line 6 in Algorithm 1 by Ziebart et al. [129]). Algorithm 4 illustrates how each time-indexed state visitation frequency is calculated.

Algorithm 4 Time-indexed State Visitation Frequency Calculation

Input: An intermediate policy π^L, initial state distribution $\mathcal{I}(s_0)$
Output: Time-indexed state visitation frequency $P_t(s|\hat{\mathbf{w}},\hat{\gamma}), \forall s \in S, \forall t \in T$

1: Initialize $P_t(s|\hat{\mathbf{w}},\hat{\gamma}) := 0$
2: **for** $s \in S$ **do**
3: $\quad P_0(s|\hat{\mathbf{w}},\hat{\gamma}) = \mathcal{I}(s_0)$
4: **end for**
5: **for** $t = 1, ..., T$ **do**
6: $\quad$ **for** $(s, a, s') \in S \times A \times S$ **do**
7: $\quad\quad P_{t+1}(s'|\hat{\mathbf{w}},\hat{\gamma}) = \sum_{(s,a)} P_t(s|\hat{\mathbf{w}},\hat{\gamma})\pi^L(a|s)P(s'|s,a)$
8: $\quad$ **end for**
9: **end for**

Algorithm 4 takes the learner's intermediate policy π^L based on $\hat{\mathbf{w}}$ and $\hat{\gamma}$, the feature value for each state $\phi(s)$, and the initial state distribution from the demonstration $\mathcal{I}(s_0)$ as input, and outputs the time-indexed state visitation frequency $P_t(s|\mathbf{w},\gamma)$, where $t \in \{0, .., T\}$. We use the empirical initial state distribution as the visitation frequency at $t = 0$ (line 2 to 4), and each subsequent time step depends on the visitation frequency of the previous time step, the policy and the transition probability (line 5 to 9).

Once we obtain the time-indexed state-action visitation frequency, Algorithm 5 shows the general learning process of obtaining $\mathbf{w}$ and γ.

Algorithm 5 Modified MaxEnt IRL Algorithm [41]

Input: $MDP \setminus R, \gamma$, demonstration set ξ, initial state distribution $\mathcal{I}(s_0)$, learning rate α
Output: Learned $\hat{\mathbf{w}}^*$ and $\hat{\gamma}^*$

1: Initialize $\hat{\mathbf{w}}$ and $\hat{\gamma}$
2: **while** not converged **do**
3: $\quad$ Compute the first terms of the gradients empirically using ξ
4: $\quad$ Solve the forward RL problem with $\hat{\mathbf{w}}$ and $\hat{\gamma}$, produce a learner's policy π^L
5: $\quad$ Compute state visitation frequencies $P_t(s|\hat{\mathbf{w}},\hat{\gamma})$ under π^L using Algorithm 4
6: $\quad$ Compute gradients $\nabla_{\hat{\mathbf{w}}}$ and $\nabla_{\hat{\gamma}}$
7: $\quad$ Update $\hat{\mathbf{w}}$ and $\hat{\gamma}$ with learning rate α
8: **end while**

The modified MaxEnt IRL algorithm takes an MDP without the reward function, the expert demonstrations and its initial state distribution as input and aims to recover the weight parameters and the discount factor γ. We first initialize the parameters, either randomly or with an educated

guess. For line 3 to 7, we first empirically compute the first terms in the two gradients (Eqs. 3.11 and 3.12) using $\hat{\mathbf{w}}$, $\hat{\gamma}$, and ξ. Next, we solve for an intermediate policy π^L and calculate the visitation frequency in line 5. We compute the gradients and update the parameters with a step size α. We repeat this process until the parameters converge.

3.4 On the Convexity of the Objective Function

With the objective function now associating with two variables $\mathbf{w}$ and γ, solving the problem is not as trivial as the original formulation by Ziebart et al. [129]. In Giwa's thesis, the author claims that the objective function is concave, and so we can directly use gradient ascent to obtain the global optimal solution. In this section, we attempt to provide a proof for this important claim. We show that, while the MaxEnt IRL with learning only the weights $\mathbf{w}$ is concave, proving the same conclusion holds when the discount factor is also an unknown variable is challenging.

First, let us examine the claim where the maximum likelihood function is concave with respect to parameter $\mathbf{w}$. Let us rewrite the sum of discounted feature of a trajectory τ in Eq. 3.9 as $\phi_\tau = \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t)$. We obtain the second derivative of the objective function w.r.t. $\mathbf{w}$ by taking the derivative of Eq. 3.9:

$$\frac{d^2}{d\mathbf{w}^2} \sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma) = 0 - \frac{\sum_\tau e^{U(\tau)} \sum_\tau \phi_\tau^2 e^{U(\tau)} - (\sum_\tau \phi_\tau e^{U(\tau)})^2}{(\sum_\tau e^{U(\tau)})^2}. \tag{3.13}$$

Let us focus on the numerator: $\sum_\tau e^{U(\tau)} \sum_\tau \phi_\tau^2 e^{U(\tau)} - (\sum_\tau \phi_\tau e^{U(\tau)})^2$. If we expand every term in the equation, for some feasible trajectory τ_1,

$$e^{U(\tau_1)} \phi_{\tau_1}^2 e^{U(\tau_1)} = (\phi_{\tau_1} e^{U(\tau_1)})^2,$$

and thus the two terms will cancel out to be 0. For any τ_2 and τ_3 where $\tau_2 \neq \tau_3$, in the first term, the multiplication is

$$e^{U(\tau_2)} \cdot \phi_{\tau_3}^2 e^{U(\tau_3)} + e^{U(\tau_3)} \cdot \phi_{\tau_2}^2 e^{U(\tau_2)} = (\phi_{\tau_2}^2 + \phi_{\tau_3}^2) e^{U(\tau_2)} e^{U(\tau_3)},$$

and in the second term

$$2(\phi_{\tau_2} \phi_{\tau_3}) e^{U(\tau_2)} e^{U(\tau_3)}.$$

We then have

$$(\phi_{\tau_2}^2 + \phi_{\tau_3}^2) e^{U(\tau_2)} e^{U(\tau_3)} - 2(\phi_{\tau_2} \phi_{\tau_3}) e^{U(\tau_2)} e^{U(\tau_3)} = e^{U(\tau_2)} e^{U(\tau_3)} (\phi_{\tau_2}^2 + \phi_{\tau_3}^2 - 2\phi_{\tau_2} \phi_{\tau_3}),$$

which is equivalent to

$$e^{U(\tau_2)} e^{U(\tau_3)} (\phi_{\tau_2} - \phi_{\tau_3})^2,$$

and it is non-negative.

Thus, for every pair of feasible τ_i and τ_j, assuming that the sum of features for each trajectory is non-zero, the numerator is always positive, making the second derivative $\frac{d^2}{d\mathbf{w}^2} \sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma)$ always negative and thus $\sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma)$ is concave w.r.t. $\mathbf{w}$. Therefore, a gradient-based method can solve the problem of learning $\mathbf{w}$ to optimality.

In Giwa's thesis [41], the author claims that when the reward function is non-negative, $R = \mathbf{w}^\top \phi \geq 0$, the objective function is concave with respect to $\mathbf{w}$ and γ as the Hessian of the objective function is negative semi-definite. Since the author did not provide a proof to this claim, in the rest of this section, we attempt to show the definiteness of the Hessian.

For the Hessian matrix to be negative semi-definite, the eigenvalues of the matrix need to be non-positive. We know the Hessian is in the form of

$$H = \begin{bmatrix} \nabla_{\mathbf{ww}} & \nabla_{\mathbf{w}\gamma} \\ \nabla_{\gamma\mathbf{w}} & \nabla_{\gamma\gamma}. \end{bmatrix}$$

The first term $\nabla_{\mathbf{ww}}$ corresponds to Eq. 3.13, which is non-positive. However, determining the signs of the other terms in the Hessian is not as trivial. Consider the second derivative of the objective function w.r.t. to γ:

$$\begin{aligned} \nabla_{\gamma\gamma} = \frac{1}{|\xi|} \sum_{t=0}^{|\tau|} t(t-1)\gamma^{t-2} \mathbf{w}^\top \phi(s_t) - \\ \frac{Z(\sum_\tau e^{U(\tau)} (\sum_{t=0}^{|\tau|} t\gamma^{t-1} \mathbf{w}^\top \phi(s_t))^2 + \sum_\tau e^{U(\tau)} \sum_{t=0}^{|\tau|} t(t-1)\gamma^{t-2} \mathbf{w}^\top \phi(s_t))}{Z^2} + \\ \frac{(\sum_\tau e^{U(\tau)} \sum_{t=0}^{|\tau|} t\gamma^{t-1} \mathbf{w}^\top \phi(s_t))^2}{Z^2} \end{aligned} \tag{3.14}$$

We can see that the sign of the derivative depends on both the demonstrations (i.e., the first term) and the expected value that involves the partition function Z (i.e., the second and the third terms). Similar results can be obtained for $\frac{d^2}{d\gamma \mathbf{w}}$ and $\frac{d^2}{d\mathbf{w}\gamma}$:

$$\begin{aligned} \frac{d^2}{d\gamma \mathbf{w}} \sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma) &= \frac{d^2}{d\mathbf{w}\gamma} \sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma) \\ &= \frac{1}{|\xi|} \sum_{t=0}^{|\tau|} t\gamma^{t-1} \phi(s_t) - \\ &\quad \frac{Z(\sum_\tau e^{U(\tau)} (\sum_{t=0}^{|\tau|} t\gamma^{t-1} \phi(s_t)) + \sum_\tau e^{U(\tau)} \sum_{t=0}^{|\tau|} (t\gamma^{t-1} \mathbf{w}^\top \phi(s_t)) \cdot \gamma^t \phi(s_t))}{Z^2} + \\ &\quad \frac{\sum_\tau e^{U(\tau)} \sum_{t=0}^{|\tau|} t\gamma^{t-1} \phi(s_t) \cdot \sum_\tau e^{U(\tau)} \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t)}{Z^2} \end{aligned} \tag{3.15}$$

Further, to determine definiteness of the Hessian, we need to compute the matrix's eigenvalue, λ, by

$$\begin{vmatrix} \nabla_{\mathbf{ww}} - \lambda & \nabla_{\mathbf{w}\gamma} \\ \nabla_{\gamma\mathbf{w}} & \nabla_{\gamma\gamma} - \lambda \end{vmatrix} = 0.$$

Solving for λ, we get

$$\lambda = \frac{\nabla_{\mathbf{ww}} + \nabla_{\gamma\gamma} \pm \sqrt{(\nabla_{\mathbf{ww}} + \nabla_{\gamma\gamma})^2 - 4(\nabla_{\mathbf{ww}} \nabla_{\gamma\gamma} - \nabla_{\mathbf{w}\gamma} \nabla_{\gamma\mathbf{w}})}}{2}.$$

To claim H is negative semi-definite is to claim that both eigenvalues are non-positive. For this to be true, we need

1. $\nabla_{\mathbf{ww}} + \nabla_{\gamma\gamma}$ to be negative, as a positive value would not lead to two negative eigenvalues.
2. $\nabla_{\mathbf{ww}}\nabla_{\gamma\gamma} - \nabla_{\mathbf{w}\gamma}\nabla_{\gamma\mathbf{w}}$ to be strictly positive. If this term is 0, we would have $\lambda = 0$; if it is negative, we would have one positive and one negative eigenvalue.

For the first condition, we know that $\nabla_{\mathbf{ww}}$ is non-positive but $\nabla_{\gamma\gamma}$ is inconclusive; for the second condition, it is more difficult to determine the sign with the terms interacting with each other. Therefore, without knowledge of the demonstrations, we do not think it is possible to conclude the convexity of the objective function w.r.t. $\mathbf{w}$ and γ jointly with the only assumption being the reward is non-negative. Moreover, one may consider an extreme case where $\mathbf{w}$ is approaching 0, this assumption would make $\nabla_{\gamma\gamma}$ effectively become 0, rendering the second condition infeasible. However, $\mathbf{w}$ being 0 is a trivial solution to the inverse problem and is generally avoided.

Since we cannot prove the claim that, in general, the objective function's Hessian being negative semi-definite directly, let us focus on something that is more manageable - the discounted sum of reward $U(\tau) = \sum_{t=0}^{|\tau|} \mathbf{w}^\top \gamma^t \phi(s_t)$.

Let us first derive the second derivatives of the function $U(\tau)$:

$$\begin{aligned}
\nabla_{\mathbf{ww}} U(\tau) &= 0 \\
\nabla_{\mathbf{w}\gamma} U(\tau) = \nabla_{\gamma\mathbf{w}} U(\tau) &= \sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t) \\
\nabla_{\gamma\gamma} U(\tau) &= \sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top \gamma^{t-2}\phi(s_t)
\end{aligned}$$

Thus, the Hessian of $U(\tau)$, H^U, is then

$$H^U = \begin{bmatrix} 0 & \sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t) \\ \sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t) & \sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top\gamma^{t-2}\phi(s_t) \end{bmatrix}$$

To obtain the eigenvalues of Hessian H^U, we set its determinant to zero:

$$\begin{vmatrix} 0-\lambda^U & \sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t) \\ \sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t) & \sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top\gamma^{t-2}\phi(s_t) - \lambda^U \end{vmatrix} = 0$$

Thus,

$$-\lambda^U(\sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top\gamma^{t-2}\phi(s_t) - \lambda^U) - (\sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t))^2 = 0$$

Similar to the previous procedure, we solve the equation above for λ^U:

$$\lambda^U = \frac{\sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top\gamma^{t-2}\phi(s_t) \pm \sqrt{(\sum_{t=2}^{|\tau|} t(t-1)\mathbf{w}^\top\gamma^{t-2}\phi(s_t))^2 + 4(\sum_{t=1}^{|\tau|} t\gamma^{t-1}\phi(s_t))^2}}{2}$$

For any $x \in \mathbb{R}$, we have $x < \sqrt{x^2+y^2}$ for some $y > 0$. Therefore, the two eigenvalues of H^U do not share the same sign, indicating that the function $U(\tau) = \sum_{t=0}^{|\tau|} \mathbf{w}^\top\gamma^t\phi(s_t)$ is neither convex nor concave.

Therefore, we can conclude that $e^{U(\tau)}$ is neither convex nor concave. In general, we cannot

conclude the convexity of the sum of non-convex functions, such as the partition function, $Z = \sum_\tau e^{U(\tau)}$. However, we do believe that it is unlikely that Z and its logarithm are both concave.

In summary, for the two components in objective function (Eq. 3.8), $\sum_{\tau\in\xi} U(\tau)$ and $|\xi| \log Z$, we can only determine that the former is neither convex nor concave while the latter is inconclusive when learning $\mathbf{w}$ and γ jointly. Further, we also cannot conclude whether the objective function is concave w.r.t. γ, because the sign of $\nabla_{\gamma\gamma}$ depends on the demonstrations' values. Therefore, while using direct gradient ascent is still a valid method, it may lead to solutions that are only locally optimal.

3.5 Discussion

So far in this chapter, the probability of a trajectory $P(\tau)$ is expressed as:

$$P(\tau) = \frac{e^{U(\tau)}}{Z}. \tag{3.16}$$

Alternatively, since we know the learned policy π^L and the state-action pairs in a given trajectory, it is intuitive to represent the probability as

$$P(\tau) = \prod_{t=0}^{|\tau|-1} \pi^L(a_t|s_t) = \prod_{t=0}^{|\tau|-1} e^{Q(s_t,a_t)-V(s_t)} \tag{3.17}$$

where $\pi^L(a_t|s_t) = e^{Q(s_t,a_t)-V(s_t)}$ since we use soft VI [129]. In the literature, Eq. 3.17 is sometimes used to replace Eq. 3.16 [106, 12] as the objective function or used as an evaluation metric [129].

One advantage of using Eq. 3.17 is that it is differentiable because of soft VI, and so we can avoid computing Z or the approximation of Z using visitation frequency and derive the gradients directly from the reward and value functions.

Intuitively, given the definition of $P(\tau)$, the probability of a trajectory occurring under the learned policy, one may think that Eqs. 3.16 and 3.17 are equivalent, namely,

$$P(\tau) = \frac{e^{U(\tau)}}{Z} = \prod_{t=0}^{|\tau|-1} \pi^L(a_t|s_t) \tag{3.18}$$

However, for Eq. 3.18 to be true, the learned policy needs to be non-stationary, meaning that the action distribution at a particular state changes over time. The condition is not often mentioned in the literature. In fact, Snoswell et al. [100] point out that many studies, including theirs, treat the learned reward function as suitable for a stationary policy since it is easier to evaluate the performance of a stationary policy computationally. However, since the reward at each time step is discounted by γ^t and $P(\tau)$ is proportional to the total discounted reward, we can interpret all feasible trajectories in Z as being produced by a non-stationary policy [100].

To show how the policy's stationarity affects Eq. 3.18, let us consider a three-state example illustrated in Figure 3.2.

In this example, each node is a state, with node 3 being the terminal and an absorbing state, and the value of each arc represents the immediate reward of choosing action j ($j \in \{1, 2, 3\}$) at state i ($i \in \{1, 2, 3\}$). Assume that the reward R^L and discount factor γ^L in the example are the optimal

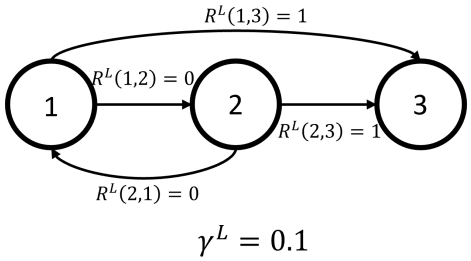


Figure 3.2: A three-state example.

parameters recovered using the modified MaxEnt IRL algorithm. We can manually perform soft VI to obtain a policy π. The final values and the action distribution for each state are shown in Table 3.2.

Table 3.2: State-action values, state values, and policy for each state in the example.

State	Q Function (Q(s,a))	V Function (V(s))	Policy ($\pi(a\|s)$)
1	Q(1, 3) = 1 Q(1, 2) = 0.135	1.35	$\pi(3\|1) = 0.705$ $\pi(2\|1) = 0.295$
2	Q(2, 3) = 1 Q(2, 1) = 0.135	1.35	$\pi(3\|2) = 0.705$ $\pi(1\|2) = 0.295$

For simplicity, let us assume that the starting state is node 1 and trajectory length is 2, that is, $\mathcal{I}(s_0 = 1) = 1$ and $|\tau| = 2$. We have three different trajectories:

$$\tau_1 = \{(1, 3), (3, 3)\}$$
$$\tau_2 = \{(1, 2), (2, 3)\}$$
$$\tau_3 = \{(1, 2), (2, 1)\}$$

We can then calculate $P(\tau_i)$ $(i \in \{1, 2, 3\}))$ using the two different methods, shown in Table 3.3.

Table 3.3: Probability of the three trajectories calculated using the two methods.

Trajectory	exp(U(τ))/Z	$\prod \pi(a\|s)$
τ_1	0.564	0.705
τ_2	0.229	0.208
τ_3	0.207	0.087

We can see that although the two trajectory distributions follow the same trend (i.e., τ_1 has the highest probability since it has the highest reward, then τ_2 and τ_3), the distribution produced by following Eq. 3.17 is less skewed and has similar probability for τ_2 and τ_3, which can be counterintuitive since τ_3 has 0 reward.

Therefore, the results shown in Table 3.3 indicate that Eq. 3.18 is invalid when the policy is stationary. Next, using the same example in Figure 3.2, we show that we can also interpret γ as an indicator of how the reward deteriorates over time in a non-stationary and undiscounted environment. Let us rename γ to β to avoid any confusion, and we define a non-stationary reward

function $R_t(s,a) = \beta^t \mathbf{w}\phi(s,a)$. Thus, the sum of rewards of a trajectory τ becomes

$$U'(\tau) = \sum_{t=0}^{|\tau|-1} \beta^t \mathbf{w}^\top \phi(s,a).$$

Although, mathematically, $U'(\tau)$ is equivalent to $U(\tau)$, they are from two different environments - the former is from an undiscounted environment (i.e., $\gamma' = 1$), while the latter is discounted. Since β^t is now part of the reward function, $P(\tau)$ now represents the probability of trajectory τ given a non-stationary policy $\hat{\pi}$ for a non-stationary environment. We illustrate this new environment with the rewards discounted by β^t at each time step in Figure 3.3 where the trajectory length is two.

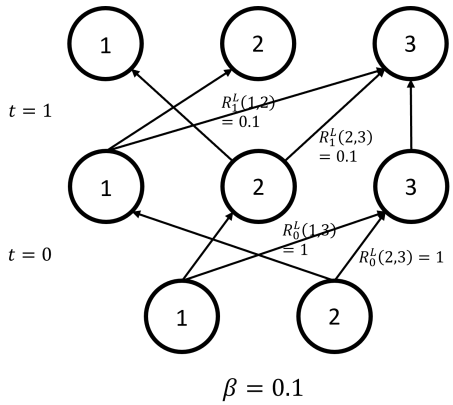


Figure 3.3: Updated example with decaying reward with length of trajectory set to two.

In Figure 3.3, we expand the original example into three layers, and the arcs connecting the layers represent the actions taken and only the ones towards node 3 have non-zero values. Note that when $t = 1$, the rewards become $\beta^1 1 = 0.1$. We then apply soft VI on this new example with time-varying reward, the resulting values and policy are shown in Table 3.4.

Table 3.4: State-action values, state values, and the corresponding non-stationary policy given time-varying rewards.

State	Time					
	t = 0			t = 1		
1	$Q_0(1,3) = 1$ $Q_0(1,2) = 0.744$	$V_0(1) = 1.573$	$\pi_0(3\|1) = 0.564$ $\pi_0(2\|1) = 0.436$	$Q_1(1,3) = 0.1$ $Q_1(1,2) = 0$	$V_1(1) = 0.744$	$\pi_1(3\|1) = 0.525$ $\pi_1(2\|1) = 0.475$
2	$Q_0(2,3) = 1$ $Q_0(2,1) = 0.744$	$V_0(2) = 1.573$	$\pi_0(3\|2) = 0.564$ $\pi_0(1\|2) = 0.436$	$Q_1(2,3) = 0.1$ $Q_1(2,1) = 0$	$V_1(2) = 0.744$	$\pi_1(3\|2) = 0.525$ $\pi_1(1\|2) = 0.475$

Assume $\mathcal{I}(s_0 = 1) = 1$, the three trajectories' probabilities are

$$P(\tau_1) = \frac{e^{U'(\tau_1)}}{Z} = \frac{e^1}{e^1 + e^{0.1} + e^0} = \pi_0(3|1) \times \pi_1(3|3) = 0.564 \tag{3.19}$$

$$P(\tau_2) = \frac{e^{U'(\tau_2)}}{Z} = \frac{e^{0.1}}{e^1 + e^{0.1} + e^0} = \pi_0(2|1) \times \pi_1(3|2) = 0.229 \tag{3.20}$$

$$P(\tau_3) = \frac{e^{U'(\tau_3)}}{Z} = \frac{e^0}{e^1 + e^{0.1} + e^0} = \pi_0(2|1) \times \pi_1(1|2) = 0.207 \tag{3.21}$$

Under the interpretation of time-varying reward R_t, the role of the original discount factor γ is

embedded in the environment, and thus it no longer represents the decision maker's view on future reward. To separate the reward and the discount factor in $U(\tau)$, we first introduce another concept - the discounted likelihood presented in Gleave and Toyer [42]:

$$P_\gamma(\tau) = \prod_{t=0}^{|\tau|} \pi_t(a_t|s_t)^{\gamma^t} \tag{3.22}$$

Eq. 3.22 is equivalent to the likelihood function (Eq. 3.17) when $\gamma = 1$. When $\gamma < 1$, the probabilities later in the trajectory will converge towards 1 when t is large. As noted by Gleave and Toyer, the discounted likelihood is not normalized. According to Lemma 3.4 in Gleave and Toyer [42], for MaxEnt IRL, if we assume the initial state distribution $\mathcal{I}(s_0)$ is deterministic, the discounted likelihood of a trajectory $P_\gamma(\tau)$ is

$$P_\gamma(\tau) = \frac{\mathcal{I}(s_0)}{e^{V_0(s_0)}} e^{U(\tau)} \tag{3.23}$$

where $V_0(s_0)$ is the value function of the initial state s_0 at $t = 0$.

Proposition 3.5.1. *For any trajectory τ, $P(\tau)$ in Eq. 3.16 is the normalized $P_\gamma(\tau)$ in a deterministic environment with deterministic $\mathcal{I}(s_0)$.*

Proof. Recall that the partition function Z is the sum of all possible trajectories' discounted return (i.e., $Z = \sum_\tau e^{U(\tau)}$). We have one initial state, such that $\mathcal{I}(s_0) = 1$.

To normalize $P_\gamma(\tau)$, we divide each term by the sum of all $P_\gamma(\tau)$,

$$P_\gamma^{norm}(\tau) = \frac{\frac{e^{U(\tau)}}{e^{V_0(s_0)}}}{\sum_\tau \frac{e^{U(\tau)}}{e^{V_0(s_0)}}} = \frac{\frac{1}{e^{V_0(s_0)}} e^{U(\tau)}}{\frac{1}{e^{V_0(s_0)}} \sum_\tau e^{U(\tau)}} = \frac{e^{U(\tau)}}{\sum_\tau e^{U(\tau)}} = P(\tau)$$

□

Therefore, following Gleave and Toyer [42], the trajectory probability $P(\tau)$ is proportional to the discounted likelihood, namely,

$$P(\tau) = \frac{e^{U(\tau)}}{Z} \propto \prod_{t=0}^{|\tau|} \pi_t(a_t|s_t)^{\gamma^t} \tag{3.24}$$

To illustrate this proposition more concretely, let us revisit the example in Figure 3.2. This time, the reward the agents receives from the environment at each state is the same regardless of the time step (i.e., $R_t(s) = R_{t'}(s), \quad \forall t, t' \in \{0, ..., |\tau| - 1\}$) and $t \neq t'$). However, when traversing the trajectory, the reward is discounted by γ^t defined in the environment. Since the environment is stationary, to create a "non-stationary" policy, we add the time step t into the state space. We still assume a trajectory length of 2, and so the state space expands from $S = \{1, 2, 3\}$ to $S = \{(1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2), (3, 1), (3, 2)\}$, where states with $t = 2$ are terminals and absorbing states.

By definition, a non-stationary policy is a policy where the action distribution of the same state might be different at different time steps due to a non-stationary environment (i.e., $\pi_t(s) \neq \pi_{t'}(s)$). When we introduce time step into the state space, the policy we obtain is still technically stationary because $s = (1, 0)$ and $s' = (1, 1)$ are two distinct states instead of state 1 being at $t = 0$ and 1.

However, for illustrative purposes, we would assume that with the newly defined state space, the optimal policy is "non-stationary". The non-terminal states' final values by soft VI are shown in Table 3.5.

Table 3.5: State-action values, state values, and the corresponding policy given the newly defined state space.

State	Q Function	V Function	Policy
(1,0)	Q(3, (1, 0)) = 1 Q(2, (1, 0)) = 0.131	1.35	$\pi(3\|(1,0)) = 0.705$ $\pi(2\|(1,0)) = 0.295$
(1,1)	Q(3, (1, 1)) = 1 Q(2, (1, 1)) = 0	1.31	$\pi(3\|(1,1)) = 0.733$ $\pi(2\|(1,1)) = 0.267$
(2,0)	Q(T, (2, 0)) = 1 Q(1, (2, 0)) = 0.131	1.35	$\pi(3\|(2,0)) = 0.795$ $\pi(1\|(2,0)) = 0.295$
(2,1)	Q(T, (2, 1)) = 1 Q(1, (2, 1)) = 0	1.31	$\pi(3\|(2,1)) = 0.733$ $\pi(1\|(2,1)) = 0.267$

Therefore, the discounted likelihood of the same three trajectories starting from node 1 are $P_\gamma(\tau_1) = 0.705$, $P_\gamma(\tau_2) = 0.295 \times 0.733^{0.1} = 0.286$, and $P_\gamma(\tau_3) = 0.295 \times 0.267^{0.1} = 0.259$, respectively.

Normalizing the discounted likelihoods, we get

$$P_\gamma^{norm}(\tau_1) = \frac{0.705}{0.705 + 0.286 + 0.259} = 0.564 \tag{3.25}$$

$$P_\gamma^{norm}(\tau_2) = \frac{0.286}{0.705 + 0.286 + 0.259} = 0.229 \tag{3.26}$$

$$P_\gamma^{norm}(\tau_3) = \frac{0.259}{0.705 + 0.286 + 0.259} = 0.207 \tag{3.27}$$

which are exactly the values shown in Table 3.3.

3.5.1 Summary

In this section, we start with an observation that in the literature, Eqs. 3.16 and 3.17 are sometimes used interchangeably. However, the condition under which these two equations are equivalent is rarely mentioned: the learned policy is non-stationary. Following this observation, we show that the expression $U(\tau) = \sum_0^{|\tau|} \gamma^t \mathbf{w}^\top \phi(s_t)$ can be interpreted in two different ways through a simple three-state example in Figure. 3.2:

1. $U(\tau)$ can be seen as the sum of undiscounted but decaying reward of the trajectory. Here, we treat $\gamma^t \phi(s_t, a_t)$ as the feature value the agents gets at the same state but different time step. Thus, we rewrite γ as β to indicate that it no longer serves as the function of a discount factor in the MDP but as part of the reward signal the environment offers. As a result, the parameter does not represent how much the agent values the future reward anymore.

2. A second interpretation is related to the concept of discounted likelihood presented in Gleave and Toyer [42]. We separate the immediate reward the agent gets and the discount factor γ. Under this interpretation, the reward signal from the environment is constant for each state across all time steps, but in calculating the return of the trajectory, the rewards are discounted by γ^t accordingly.

These two interpretations regarding what is being discounted in $U(\tau)$ are generally not discussed in the literature, potentially because almost all IRL studies focus on learning the weight parameters only and the discount factor γ is hidden in the formulation. However, since our goal is to learn the discount factor as well, the two interpretations above lead to different policies as shown in Tables 3.4 and 3.5. We believe that the latter is more suitable to the definition of discount factor in an MDP and so $P(\tau) \propto \prod_{t=0}^{|\tau|} \pi_t(a_t|s_t)^{\gamma^t}$ (Eq. 3.24). Thus, when computing $P(\tau)$, even with a non-stationary policy, the product of probabilities of actions in a trajectory is only an approximation of the true probability. To obtain the accurate probability of a trajectory, we need to enumerate all feasible trajectories for both Z and $\sum_\tau \prod_{t=0}^{|\tau|} \pi_t(a_t|s_t)^{\gamma^t}$, which is only possible for small state and action spaces. Further study is required to investigate and establish a more efficient way to calculate or estimate the partition function.

Finally, although it is clear that the policy corresponding to the learned reward function under the MaxEnt IRL framework is non-stationary, as mentioned before, many studies assume the policy is stationary since it is easier to compute [100]. In this thesis, we also choose to use a stationary policy since the environment in our application presented in the next chapter is static. As noted previously, this simplification has several limitations, one of which being that because we depend on the learned policy in Algorithm 4, the visitation frequencies of all states are approximations of the true frequencies.

3.6 Conclusion

Learning the discount factor γ along with the weight parameters is rare in the field of IRL since γ is often treated as a mathematical convenience for convergence criteria. However, we can also interpret γ as an indicator of how much the decision maker values the future rewards, and thus has an impact on the decision maker's behaviour. In this chapter, we present an algorithm based on the MaxEnt IRL framework proposed by Ziebart et al. [129], which enables the learning of γ and the weights. The proposed modified MaxEnt IRL algorithm is built upon the work by Giwa [40, 41], who first proposed learning γ and the weights simultaneously. However, we point out several errors in Giwa's formulation and present an updated algorithm. We then investigate the claim about the concavity of the objective function when learning the two components in the MDP jointly, and conclude that we cannot determine the objective function is concave as the author claimed. Finally, we provide some insight into different ways of expressing the trajectory probability, $P(\tau)$, and we show that the standard equivalence seen in the literature does not hold.

Chapter 4

Modelling Wild Vervet Monkey Behaviour When Foraging Alone

4.1 Introduction

In this chapter, we apply the methods presented in the previous chapter to explaining wild vervet monkeys' foraging behaviour. We believe that on top of recovering the reward function, learning the discount factor is also particularly applicable to behavioural studies. As we have established in Chapter 3, the discount factor can be interpreted as how much the decision maker values the future reward, a factor that changes the choices that it makes. Rather than making assumptions about whether our subjects are myopic or not, treating the discount factor as an unknown to be recovered from the observed behaviours may offer valuable insight on their decision-making process.

As mentioned in Chapter 2, animal behaviour is a significantly less popular application of inverse reinforcement learning (IRL) than others such as robotics or human behaviour. Although there have been several studies in the literature on animal behaviour [118, 48, 13], the majority of these studies were conducted in a laboratory setting, where data can be collected in a controlled environment. Existing work on wild animal is very limited, and to our knowledge, there are only a few studies [49, 85]. Data collection in the wild is more challenging due to the unpredictability of animal behaviour. Thus, researchers often use aids such as GPS to help more accurately track the animals' movements [49, 85]. In the study that we base this chapter on, however, the experimental data were manually recorded by human experts, an additional challenge as it can lead to incomplete data, potentially hindering our modelling capability.

The foraging experiment was conducted by Arseneau-Robar et al. [10], where one of the main findings is that when the monkeys forage alone, they tend to maximize their total reward while minimizing their travel distance, with which our experimental results align; and when foraging in competition, monkeys tend to prioritize the preferred reward while responding to the social context. In this chapter, we focus on modelling the behaviours when foraging alone. Some key research questions are:

1. How much does the monkey care about the food reward versus the travel distance?
2. When there is no competition, is the subject myopic or farsighted?

3. Does the starting position in the foraging experiment affect the monkey's decisions? If so, how different is the behaviour?

This chapter is organized as follows. We first introduce the foraging experiment setup and describe some key characteristics and assumptions we made in Section 4.2. Next, we present the Markov decision process (MDP) model that represents the foraging experiment and our approach to solving the IRL problem in Section 4.3. We then present the experimental results in Section 4.4 followed by a discussion on the problem and the challenges of applying IRL to our specific application in Section 4.5. We propose some alternative modelling options if the data required is available in Section 4.5. Finally, we conclude this chapter in Section 4.6.

4.2 The Foraging Experimental Setup

The foraging experimental setup in Arseneau-Robar et al. [10] is described as follows. There were five platforms placed 5 meters apart from each other and arranged in a pentagon. Four of the five platforms were baited with three corn kernels, and the remaining platform was baited with half of a banana placed in a container. Figure 4.1 illustrates one possible setup.

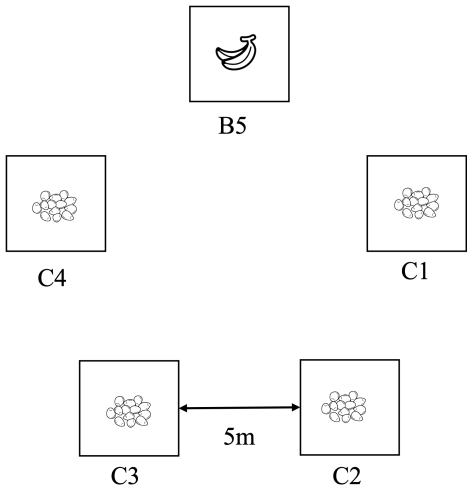


Figure 4.1: Foraging experimental set-up.

The experimental site was set in an open area near the habitat of a group of monkeys, and thus the monkeys can enter or exit the trial freely. The monkey that participated in a trial is referred to as the "focal" monkey. Although various types of information were collected for each trial, our process of modelling the experiment as an MDP only utilizes two specific pieces of information: the sequence of platforms visited and which platform was baited with the banana. We choose to focus on the decision sequence and the banana platform because we believe they are the most unbiased and accurate information in a dataset that was manually collected by human experts [10]. Due to the unpredictable nature of wild animal behaviour, without the proper aid of technology, manual data collection may result in incomplete data. For example, the duration of retrieving the banana from the container was documented in each trial, but the time spent travelling between platforms or the time spent on corn platforms was not. Therefore, if we want to discretize the state space by

time steps, we would need to make additional assumptions about the time it takes for the focal to complete each task, which may be biased and can lead to a decrease in data quality. Additional information about each trial includes: the focal's sex, age, and rank in the group; the closest platform to the focal before the foraging trial began; whether there was an "audience" of the trial, that is, monkey(s) that observed the trial from a distance but did not participate.

4.3 Methodologies

In order to study the monkeys' foraging behaviours using IRL, we first model the observed sequential decision-making process as an MDP, where the focal is the agent. The global assumptions/modifications made about the foraging experiments are described below.

1. Although the banana platform was randomly chosen in each trial, we set the platform label in each trial to be C1 to B5 as shown in Figure 4.1, making the last platform, platform 5, the banana platform, and platform 1 to 4 the corn platforms.
2. All five platforms are properly baited before the focal enters the trial.
3. Once the focal visits a platform, we assume that the food on that platform is taken.
4. We assume that travelling from one platform to another and consuming the food on each platform do not take any time.
5. We assume that the audience of the trial (if there is any) does not affect the behaviour of the focal.
6. All trials are completed, namely, the focal collected all available food in the site.
7. We assume that each trajectory starts with the focal being on a platform.

We make Assumption 1 to ensure data clarity and ease of interpretation. This assumption does not alter the actual decision sequence because the pentagon's symmetrical structure allows us to think of the process of transforming any platform label as rotating the pentagon until the original banana platform aligns with B5 in Figure 4.1. The focal's starting platform and path are then similarly transformed. With such a transformation, we implicitly assume that the monkey's behaviour of starting from C3 (or C4) is the same as C2 (or C1), which may be a potential source of error. Assumption 4 is derived from the fact that, as mentioned in Section 4.2, the data only contains the duration of a specific task but lacks information on others. Further, in the original study, the trial begins before the focal reaches a platform, as the focal's first choice is an important aspect in Arseneau-Robar et al. [10]. However, in Assumption 7, our focus shifts to the sequence of decisions made by the focal following the first decision, as we do not have enough information to model the first decision in the MDP. Based on these assumptions, we now present the MDP model for representing the foraging experiment.

4.3.1 Modelling Foraging Alone as an MDP

We denote the MDP model that considers the case when the focal forages alone as MDP-NoComp. The known components of MDP-NoComp, $\langle S, A, P_{sa} \rangle$, are described as follows.

- **State** S: we discretized the foraging experiment by platforms and each state consists of two elements:
  - `f_loc`: the platform the focal is currently on. `f_loc` $\in$ {C1, C2, C3, C4, B5}.
  - `remaining_plats`: a binary 5-tuple in which an element takes the value 1 if the platform has not been visited by the focal. For example, if platform C1 is the only remaining platform, `remaining_plats` $= (1, 0, 0, 0, 0)$.

  Whichever platform the focal is on, the food is assumed to be taken (Assumption 3). Therefore, for each `f_loc`, there are 16 possible combinations of `remaining_plats`. In total, there are 80 states in the state space S. States with `remaining_plats` $= (0, 0, 0, 0, 0)$ are considered terminal states.

- **Action** A: an action $a \in$ {C1, C2, C3, C4, B5} represents the focal's choice of moving to another platform ($a \neq$ `f_loc`) or staying at the platform that it is currently on ($a =$ `f_loc`). We allow the focal to go back to a visited platform in this model to be more general, although it is highly unlikely that a monkey would do so and this behaviour was never observed in the data.

- **Transition Probability** P_{sa}: The transition probabilities are deterministic in the model since an action a represents the focal's decision that we can observe (i.e., physically moving to a platform) rather than its internal thought process when making the decision (i.e., *wanting to* move to a platform). The latter would lead to potentially stochastic dynamics since the focal may want to move to a certain platform but end up on a different one. However, since we do have information on the monkey's internal decisions, we define the transition probabilities to be deterministic. Under P_{sa}, the agent transitions into the next state $s' = (\texttt{f\_loc}', \texttt{remaining\_plats}')$ with probability 1 where $\texttt{f\_loc}'$ depends on a and $\texttt{remaining\_plats}' = \texttt{remaining\_plats} \setminus \texttt{f\_loc}'$. For example, for $s = (\text{C1}, (0, 1, 1, 1, 1))$ and $a = \text{B5}$, s' is $(\text{B5}, (0, 1, 1, 1, 0))$.

Based on the observations in Arseneau-Robar et al. [10], we expect that, when foraging alone, the focal not only considers the food reward but also the distance it needs to travel in order to get the reward. Thus, we define two features for each state-action pair in this model:

- **Food**: According to the two-choice experimental preference trials ran by Arseneau-Robar et al. [10], the monkeys prefer the banana 20 times over the dried corn kernels. Thus, the feature value of a state-action that leads to a corn platform is set to 0.05, and 1 to the banana platform. If the focal decided to go back to a platform that it has already visited or stay at the current platform, the value is 0.

- **Distance**: Given the pentagon-shaped foraging array, the feature value of moving to an adjacent platform is 5 meters, and moving to a non-adjacent platform is 8.09 meters.

Following the common approach in IRL literature, we set the immediate reward function at each state-action pair to be a linear combination of the weighted features, namely,

$$R(s, a) = w_1 \texttt{Food}(s, a) - w_2 \texttt{Distance}(s, a),$$

where w_1 and w_2 are the weights of each feature that we aim to learn. They represent how much the focal cares about the food and about the distance it has to travel to obtain it.

Further, the model is flexible in what actions the focal can take at each state. Thus, the immediate reward of taking action a in a non-terminal state s is:

$$R(s,a) = \begin{cases} w_1\texttt{Food}(s,a) - w_2\texttt{Distance}(s,a) & \text{if going to a platform with food} \\ -w_2\texttt{Distance}(s,a) & \text{if going to a platform with no food} \\ -\rho & \text{if staying at the current platform} \end{cases} \tag{4.1}$$

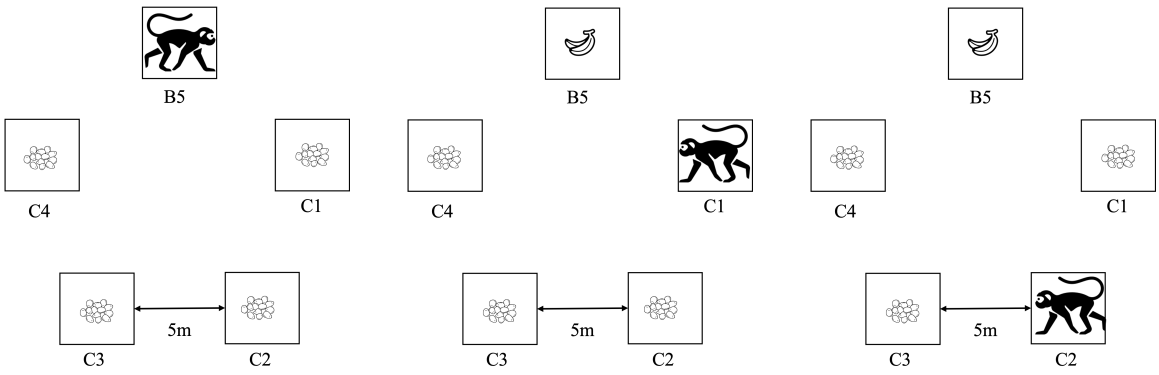
where $\rho \in \mathrm{R}^+$. We assign a negative constant for staying at the same platform because in the given demonstrations, we observed that when foraging alone, the focal never chose to stay at a platform during a trial.

In addition to the weights, we are also interested in learning the discount factor γ, which represents how much the focal values the future reward. In Chapter 3, we define the sum of discounted reward of a given trajectory τ as:

$$R(\tau) = \mathbf{w}^\top \sum_{t=0}^{|\tau|-1} \gamma^t \phi(s_t, a_t)$$

where $\mathbf{w} = (w_1, w_2)$ and $\phi(s_t, a_t)$ is the feature vector of the t^{th} state-action pair in τ. We are interested in finding the optimal combination of $(\mathbf{w}^*, \gamma^*)$ that best explains the observed data.

Recall that in Assumption 7, we state that the foraging trial begins with the focal already being on a platform. Therefore, we set three different starting states - $s_0 \in \{$(B5, $(1,1,1,1,0)$), (C1, $(0,1,1,1,1)$), (C2, $(1,0,1,1,1)$)$\}$, as shown in Figure 4.2. Note that we only have three instead of five starting states because of the symmetrical structure of a pentagon, namely, starting from C4 is equivalent to starting at C1, and C3 equivalent to C2. Thus, we can convert trajectories starting from C3 or C4 to the appropriate starting platforms and increase the number of available trajectories in each group.



(a) Starting from platform B5. (b) Starting from platform C1. (c) Starting from platform C2.

Figure 4.2: Three possible starting states.

4.3.2 Recovering the Parameters Using IRL

To learn $\mathbf{w}$ and the discount factor γ, we use the modified MaxEnt IRL framework presented in Chapter 3, where, given a set of demonstrations ξ, we solve the maximum log likelihood estimation (MLE) problem with respect to the two parameters. Namely,

$$\mathbf{w}^*, \gamma^* = \arg\max_{\mathbf{w},\gamma} \sum_{\tau \in \xi} \log P(\tau|\mathbf{w}, \gamma). \tag{4.2}$$

In this study, we employ basic gradient ascent methods to solve the MLE problem and report the best $\mathbf{w}^*$ and γ^* found by using the gradients Eq. 3.11 and 3.12, and Algorithm 5 presented in Section 3.3. In addition to learning the parameters $\mathbf{w}$ and γ jointly, we also try learning each parameter separately in an iterative fashion, presented in Algorithm 6. The main motivation behind using Algorithm 6 is that although the convexity of the objective function (Eq. 4.2) is inconclusive as presented in Section 3.4, we know that Eq. 4.2 is concave w.r.t. $\mathbf{w}$ and may be convex w.r.t. γ. Therefore, solving the two parameters separately may be more likely to lead us to a better solution than learning the two parameters simultaneously.

The iterative approach offers an additional benefit for the optimization problem and is related to the concept of block coordinate ascent (BCA) [113]. In Algorithm 6, the discount factor γ and the weight $\mathbf{w}$ can be seen as two blocks of coordinates and we optimize the objective function w.r.t. one block by fixing the value of the other. Although our initial intention for using Algorithm 6 was not to solve the optimization problem more efficiently, using BCA leads to two lower-dimensional subproblems (i.e., a 1D problem w.r.t. γ and a 2D problem w.r.t. $\mathbf{w}$) that are easier to solve than the full problem 3D problem.

Moreover, since our reward function is low-dimensional, we can further divide $\mathbf{w}$ into w_1 and w_2 and solve the objective function w.r.t. γ, w_1, and w_2 iteratively, resulting in three one-dimensional subproblems. When optimizing 1D problems, there are various search algorithms that can identify the local optimum, such as bracketing [62]. However, these algorithms require us to calculate the objective function value with specific parameters, which is not trivial since we need to compute the partition function Z. Thus, while coordinate ascent leads to simpler and lower-dimensional subproblems, it may not provide any computational benefit.

Algorithm 6 is slightly modified from Algorithm 5 and is similar to the one presented in Ashwood et al. [12]. In each iteration i, we first update the weight parameters $\mathbf{w}$ (line 3 to 7), and then we use the updated weight to update the discount factor γ with the new weights (line 8 to 12). Note that, because we are solving $\mathbf{w}$ and γ iteratively, we need to solve the forward problem within each inner loop j, thus, in total, we need to solve the forward problem $2MN$ times.

4.4 Experiment

4.4.1 Data

The dataset used in Arseneau-Robar et al. [10] contains 782 trials in which the focal foraged without competition. There are some incomplete trials due to various factors, such as the focal not completing the experiments. We remove those trials and 753 records remain.

The raw data is stored in a text-based format that contains the sequence of decisions made by

Algorithm 6 Iterative Modified MaxEnt IRL Algorithm

Input: $MDP \setminus R, \gamma$, demonstration set ξ, initial state distribution $\mathcal{I}(s_0)$, learning rate α
Output: Learned $\hat{\mathbf{w}}^*$ and $\hat{\gamma}^*$

Initialize $\hat{\mathbf{w}}$ and $\hat{\gamma}$
for $i = 1, ..., N$ **do**
  for $j = 1, ..., M$ **do**
    Solve the forward RL problem with $\hat{\mathbf{w}}_j^i$ and $\hat{\gamma}^i$, produce a learner's policy π^L.
    Update $\hat{\mathbf{w}}^i$ by computing the gradient $\nabla \mathbf{w}_j$ using Eq. 3.11: $\hat{\mathbf{w}}_{j+1}^i = \hat{\mathbf{w}}_j^i + \alpha_{\mathbf{w}} \nabla \mathbf{w}_j$
  end for
  Set $\hat{\mathbf{w}}^{i+1} = \hat{\mathbf{w}}_M^i$
  for $j = 1, .., M$ **do**
    Solve the forward RL problem with $\hat{\mathbf{w}}^{i+1}$ and $\hat{\gamma}^i$, produce a learner's policy π^L
    Update $\hat{\gamma}^i$ by computing the gradient $\nabla \gamma_j$ using Eq. 3.12: $\hat{\gamma}_{j+1}^i = \hat{\gamma}_j^i + \alpha_\gamma \nabla \gamma_j$
  end for
  Set $\hat{\gamma}^{i+1} = \hat{\gamma}_M^i$
end for

the focal. The original platform that was baited with banana was also provided. For the foraging trials with no competitor, transforming the raw data into feasible trajectories is relatively simple. An example of how we do so is illustrated in Table 4.1.

Table 4.1: Transforming the example sequence in raw data to a feasible trajectory.

Raw Data	Banana Platform	Trajectory
21543	1	{(C1, $(0,1,1,1,1)$), (B5, $(0,1,1,1,0)$, (C4, $(0,1,1,0,0)$), (C3, $(0,1,0,0,0)$), (C2, $(0,0,0,0,0)$)}

In this example, the order of platforms visited is "21543" and platform 1 was baited with banana. Recall that we fix the banana platform to be B5, and thus we shift the platform numbers accordingly.

As presented in Section 4.3, there are three starting non-symmetric platforms for the foraging alone case—C1, C2, and B5. Of the 753 trajectories in the dataset, 183 trajectories started at C1 (24.3%), 458 started at C2 (60.8%) and 112 started at B5 (14.9%). We obtained the training and test set through an 80-20 split for each subset.

4.4.2 Experimental Setup

For the computational experiments, we randomly and uniformly initialize the parameters as follows:

$$w_1 \in U[10, 11]$$
$$w_2 \in U(0, 0.1]$$
$$\gamma \in U[0.01, 0.99]$$

Note that the range of w_1 and w_2's initialization ensure that the reward of going to an adjacent platform will be non-negative while going to a non-adjacent platform will result in a negative reward except for going to the banana platform. We initialize w_2 to be positive based on the assumption that travelling between platforms is a cost.

Soft VI [129] (see Section 2.2) was used for solving the forward problem, where we write the Q

function, value function, and policy as:

$$V^{Soft}(s) = \sigma \log \sum_{a \in A} e^{\frac{Q^{Soft}(s,a)}{\sigma}}$$
$$Q^{Soft}(s,a) = R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^{Soft}(s')$$
$$\pi(a|s) = e^{\frac{Q^{Soft}(s,a) - V^{Soft}(s)}{\sigma}}.$$

The σ in the equations above represents the "temperature", and a lower temperature leads to more deterministic behaviour. While one may think that σ has a similar function as the discount factor, note that there is a distinction between the role of σ and γ: σ indicates how rational the agent is, while γ, as we have established previously, represents how myopic or farsighted the agent is regarding the future reward. In our experiments, we follow the convention in literature [128, 48, 20] and set the temperature to be 1, which can also be interpreted as assuming that the focal is not a highly rational decision maker.

We set a computational budget for each set of experiments based on how many times the visitation frequency matrix is computed, since that is the most time-consuming part in the training process. We denote the number of times that the visitation frequency matrix is built as `max_iter`, and set `max_iter` $= 60$.

There are three possible starting platforms: B5, C1, and C2. To test the hypothesis of whether starting from different platforms affects the focal's trade-off between food and travel distance, we separate the data based on starting platform and develop two learning schemes (LS):

1. LS1: We learn one set of parameters that optimize all three sets of data. In each iteration i, we use the same intermediate parameters and calculate the gradients for each dataset: $\nabla \mathbf{w}_i^k, \nabla \gamma_i^k$ where $k \in \{\text{B5, C1, C2}\}$. We then update the parameters by the weighted average of the three sets of gradients: $\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha_{\mathbf{w}} \sum_k c_k \nabla \mathbf{w}_i^k$ and $\gamma_{i+1} = \gamma_i + \alpha_\gamma \sum_k c_k \nabla \gamma_i^k$, where α represents the learning rate and c_k is the proportion of trajectories starting from platform k in the training data. We denote the learned parameters as $\mathbf{w}^{LS1}$ and γ^{LS1}

2. LS2: We train each starting platform independently and learn a set of parameters for each dataset. That is, we learn three separate sets of $(\mathbf{w}^l, \gamma^l)$ for each starting platform $l \in \{$LS2-C1, LS2-C2, LS2-B5$\}$.

For LS1, since we update the parameters based on the three gradients, three visitation frequency matrices are computed in each iteration. Therefore, we ran the modified MaxEnt IRL (Algorithm 5) for $\frac{\texttt{max\_iter}}{3} = 20$ times. For the iterative algorithm (Algorithm 6), the number of iterations for the inner loop is fixed to $M = 5$ for both learning $\mathbf{w}$ and γ, and so the N is set to 2. For LS2, we ran Algorithm 5 `max_iter` times and ran Algorithm 6 with $N = 6$ and $M = 5$. Moreover, for each experiment, we calculate the log likelihood (LL) of the intermediate parameters after each iteration, and the parameters that produce the best LL are returned at the end of the learning process along with the LL.

Through preliminary experiments, the learning rate $\alpha_{\mathbf{w}}$ and α_γ are set to [0.05, 0.02] and 0.02, respectively, for both learning schemes.

Finally, we compared the two methods with random search as a baseline. For random search, instead of updating the parameters in each iteration, we randomly sample new parameter values as

follows:

$$w_1^{Random} \in U[10, 11]$$
$$w_2^{Random} \in U(0, 1]$$
$$\gamma^{Random} \in U[0.01, 0.99]$$

4.4.3 Results

In this section, we present both the training and test results for the three methods: vanilla gradient ascent (VGA, Algorithm 5), iterative gradient ascent (IGA, Algorithm 6), and random search (RS). We apply these methods to both learning schemes, so in total, there are 12 sets of recovered parameters $(\mathbf{w}_m^{LS}, \gamma_m^{LS})$, where $LS \in$ {LS1, LS2-C1, LS2-C2, LS2-B5} and $m \in$ {VGA, IGA, RS}.

4.4.3.1 Training Results

Table 4.2 illustrates the learned parameters. Overall, the learned parameters are similar across different methods and the learned discount factors are all close to 1.

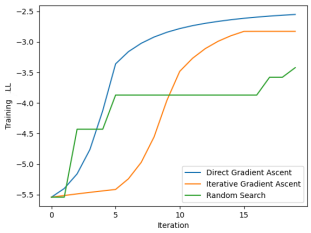
Table 4.2: Learned parameters by starting platform and methods.

Learning Scheme (LS)		Learned Parameters								
		Vanilla Gradient Ascent			Iterative Gradient Ascent			Random Search		
Temperature		w_1	w_2	γ	w_1	w_2	γ	w_1	w_2	γ
$\sigma = 1$	LS1	10.605	0.647	0.986	10.549	0.472	0.99	10.112	0.252	0.956
	LS2-C1	10.623	0.888	0.99	10.599	0.709	0.982	10.778	0.87	0.969
	LS2-C2	10.666	0.689	0.98	10.59	0.66	0.983	10.279	0.586	0.96
	LS2-B5	10.657	0.785	0.99	10.653	0.674	0.99	10.419	0.903	0.97

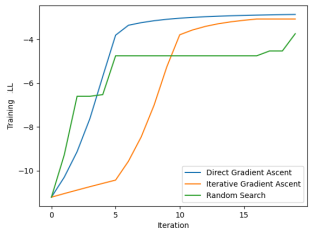
Figure 4.3 shows the best training LL at each iteration for both learning schemes. Figure 4.3a to 4.3c show the results of LS1 by starting platforms, and Figure 4.3d to 4.3f show the results for LS2. The vanilla gradient ascent method has the best overall performance in terms of achieving better LLs except for LS2-C1, where random search obtained the best LL within 10 iterations. Further, we can see that the best LLs plateaued in later iterations for both learning schemes, indicating that we arrived at good solutions early on in the training process.

Figure 4.4 shows the optimal paths of most starting platform-method pairs given their learned policies. Unsurprisingly, the best path, regardless of where the focal starts, is to go around the pentagon array to collect all the available food rewards while minimizing the travel distance. This result matches with our intuition of how the focal would behave when foraging alone and aligns with the results in Arseneau-Robar [10].

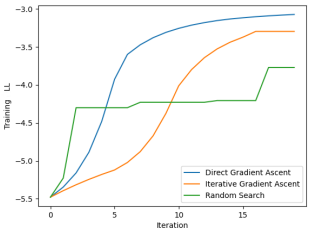
One exception is the optimal path produced by parameters $(\mathbf{w}_{RS}^{LS1}, \gamma_{RS}^{LS1})$ when starting from C2: instead of going in order, the optimal behaviour based on this set of parameters is to prioritize the banana, then collect the corn kernels on the adjacent platform C1, and instead of moving directly to C3, the action with the highest probability according to the learned policy is to go to C3 through C2 and collect the last reward at C4. This result seems to suggest that the focal would rather choose to visit the platforms in order than moving to an available platform directly even if the travel distance is longer.



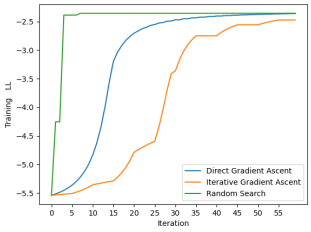
(a) Training LL using LS1 starting from C1.



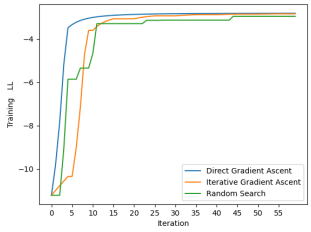
(b) Training LL using LS1 starting from C2.



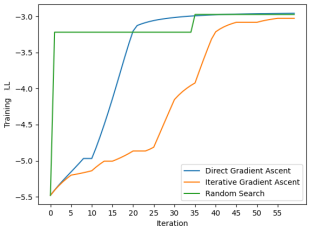
(c) Training LL using LS1 starting from B5.



(d) Training LL using LS2-C1.



(e) Training LL using LS2-C2.



(f) Training LL using LS2-B5.

Figure 4.3: Best training LL in each iteration by learning scheme and starting platform.

4.4.3.2 Test Results

We then apply the learning results to the test set and compare them using three metrics: the average test LL, the expected food feature value difference, and the expected travel distance feature difference between the test set and the learned policy.

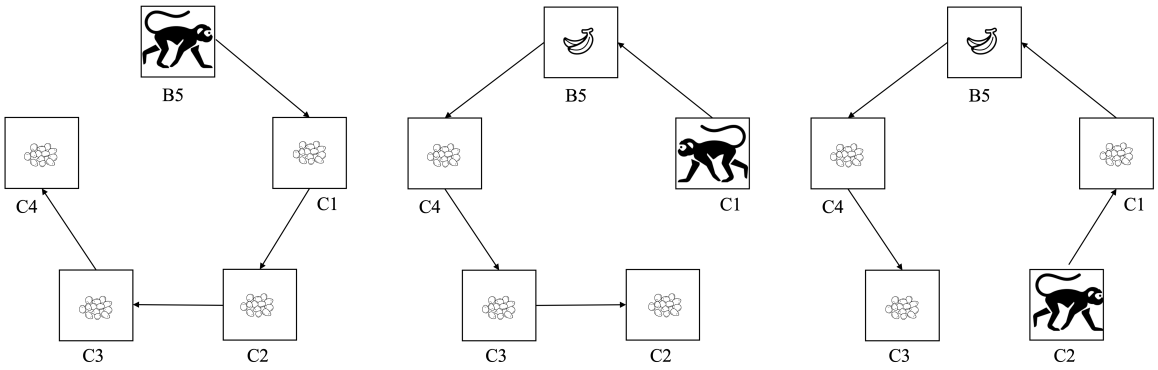
Table 4.3 shows the results of the LLs for the test set. We compute the LLs for each starting platform using $\mathbf{w}^{LS1}$ and γ^{LS1} and compare them with the LL using starting platform specific parameters. We use dashes in the table to indicate that we only computed the LLs for the test set that corresponds to the starting platform (e.g., $\mathbf{w}^{LS2-C1}$ and γ^{LS2-C1} were used to compute the result for trajectories starting from C1, not C2 and B5).

Table 4.3: Test LL based on starting platform and learned parameters.

		Test LL								
		Vanilla Gradient Ascent			Iterative Gradient Ascent			Random Search		
Temperature	Learning Scheme	C1	C2	B5	C1	C2	B5	C1	C2	B5
$\sigma = 1$	LS1	-2.672	-2.619	-3.451	-2.924	-2.897	-3.579	-3.53	-3.645	-4.105
	LS2-C1	-2.524	-	-	-2.62	-	-	-2.549	-	-
	LS2-C2	-	-2.572	-	-	-2.604	-	-	-2.748	-
	LS2-B5	-	-	-3.429	-	-	-3.432	-	-	-3.5

We can see that using starting platform specific parameters leads to better results, and vanilla gradient ascent method has the best performance, suggesting that it is a promising approach even though we do not have a convexity guarantee of the objective function.

Table 4.4 and 4.5 show the expected discounted feature count difference between the test set and the learned policies. For the food feature in Table 4.4, using starting platform specific parameters again leads to better results, aligning with the results in Table 4.3. Although the difference in Table 4.4 may seem to be very small, recall that we set the feature value for corn kernels to be 0.05 and



(a) Optimal path starting from B5.

(b) Optimal path starting from C1.

(c) Optimal path starting from C2.

Figure 4.4: Optimal paths based on starting platform.

banana to be 1. Vanilla gradient ascent has the best performance overall, where all values are less than 0.05.

Table 4.4: Expected food feature value difference in the test set by learned parameters based on LS and methods.

		Test Expected Feature Difference – Food								
		Vanilla Gradient Ascent			Iterative Gradient Ascent			Random Search		
Temperature	Learning Scheme	C1	C2	B5	C1	C2	B5	C1	C2	B5
$\sigma = 1$	LS1	0.024	0.03	0.005	0.069	0.081	0.015	0.101	0.103	0.05
	LS2-C1	0.009	-	-	0.018	-	-	0.01	-	-
	LS2-C2	-	0.024	-	-	0.028	-	-	0.032	-
	LS2-B5	-	-	0.002	-	-	0.004	-	-	0.001

For the distance feature difference shown in Table 4.5, the results show that for starting at platform B5, using $\mathbf{w}_{VGA}^{LS2-B5}$ and γ_{VGA}^{LS2-B5} obtained by vanilla gradient ascent is actually worse than using $\mathbf{w}^{LS1}$ and γ^{LS1}. Again, vanilla gradient ascent produces the best overall performance among the three methods.

Table 4.5: Expected travel distance feature value difference in the test set by learned parameters based on LS and methods.

		Test Expected Feature Difference – Travel Distance								
		Vanilla Gradient Ascent			Iterative Gradient Ascent			Random Search		
Temperature	Learning Scheme	C1	C2	B5	C1	C2	B5	C1	C2	B5
$\sigma = 1$	LS1	0.671	0.535	-0.04	1.673	1.602	0.973	2.928	3.502	2.501
	LS2-C1	-0.05	-	-	0.428	-	-	0.011	-	-
	LS2-C2	-	0.357	-	-	0.479	-	-	0.88	-
	LS2-B5	-	-	-0.544	-	-	-0.169	-	-	-0.765

4.5 Discussion

In this chapter, we applied the modified MaxEnt IRL algorithms presented in Chapter 3 to an application of animal foraging behaviour, and compare it against a similar but iterative algorithm and random search. Using these results, we can now attempt to answer the research questions posed at

the beginning of this chapter.

1. What is the trade-off between the food and travel distance?
Although the initial weights were drawn from a range that ensures that the focal would receive a non-negative reward by moving to an adjacent platform, the learned weights for distance are higher than the initial w_2 across all experiments. As a result, only moving towards B5 would have a positive reward. For example, consider $\mathbf{w}_{VGA}^{LS2-C1} = (10.623, 0.888)$, if B5 is the adjacent platform, the focal will get a reward of $10.623 \cdot 1 - 0.888 \cdot 5 = 6.813$; if B5 is non-adjacent, the reward is $10.623 \cdot 1 - 0.888 \cdot 8.09 = 3.439$. For any corn platform, the focal will get -3.909 if the platform is adjacent and -6.653 if it is non-adjacent.

This result suggests that there is no incentive (a positive reward) for the focal to get the corn. One possible interpretation could be that the focal only truly cares about the banana. When the banana is no longer available, the focal is simply visiting the remaining platforms with food by following the shortest path. This result is rather surprising because even though the optimal behaviour based on the recovered parameters matches the observations in Arseneau-Robar et al. [10], it does not match with our expectation of *why* the monkey is visiting the platforms in order or indeed why it is visiting them at all. Intuitively, we would assume that even though the corn kernels are less preferred, the focal would still value the food more and thus is motivated by maximizing the food rewards. However, our experiments imply that the monkeys do not care about the corn kernels. We tested multiple initial weight parameters and discount factors, all ensuring that moving to a corn platform would have positive reward (i.e., high w_1 and low w_2), but all converged towards points that produce negative rewards for corn. Given the concavity of the objective function w.r.t. $\mathbf{w}$, we do not think that there exists another optimal solution where moving to an available corn platform is an incentive.

However, our MDP model is a simplification of the real foraging experiment, and so there may be factors other than food and travel distance that affect the focal's decision-making process. For example, one hypothesis may be that, if there is any audience of the trial, the focal may be motivated by the fact that it does not want the others to get the food. Further, during the foraging experiment, the monkey is free to leave the site at any time without finishing the trial, which is not reflected in our MDP model. Thus, one possible explanation of why the monkey would continue foraging even with a negative reward for corn may be because the model does not have the action "leave experiment". We leave the investigation of this model modification to future work.

2. Is the focal myopic or farsighted?
When foraging alone, our results suggest that the focal tends to be a very farsighted decision maker, since the recovered discount factors are above 0.97 for all experiments. Therefore, the focal treats the future reward almost as equally as the present reward. One potential interpretation for this result may be the focal takes the future rewards into account because it understands that it can collect all available rewards since it is not under any foraging pressure.

3. Does the starting position in the foraging experiment affect the monkey's decisions?
Although the focal is a farsighted decision maker and our results suggest that the focal is most likely

to go around the array to collect all the food, which way it chooses to go around (i.e., clockwise or counterclockwise) matters depending on which platform the focal starts with.

Let us consider the three sets of learned parameters using LS2-VGA. When starting from C1, the policy based on $(\mathbf{w}_{VGA}^{LS2-C1}, \gamma_{VGA}^{LS2-C1})$ shows that the probability of moving counterclockwise to B5 is 0.562, and the probability of moving to C1 is 0.425, and a small probability of moving to either C3 or C4. This result matches with our expectation since B5 is adjacent to C1, the focal should be more likely to go collect the banana rather than collecting all the corn kernels before arriving at B5 at the end. In the real data, we also observe that while the most frequent path is {C1, B5, C4, C3, C2}, the second most frequent path is going the other way {C1, C2, C3, C4, B5}.

Similarly, when starting from C2, according to the policy based on $(\mathbf{w}_{VGA}^{LS2-C2}, \gamma_{VGA}^{LS2-C2})$, moving to C1 has a probability of 0.512, and to C3 is 0.433. While the focal is still more likely to move counterclockwise, the difference between going to C1 and C3 is smaller compared to moving to B5 or C2 from C1. This is because the difference between {C2, C1, B5, C4, C3} and {C2, C3, C4, B5, C1} is whether the focal gets the banana as the second or third reward, compared to whether it is the first or the fourth when starting from C1. Although the discount factor is very high ($\gamma = 0.99$), there still exists a slight preference to get the banana as quickly as possible since the banana is the only positive reward in the trial and the monkey does not truly value the future equally as the present (i.e., if $\gamma = 1$).

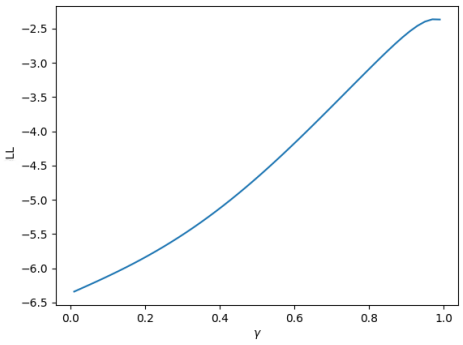
Finally, when starting from B5, since the banana has been taken, the focal should have equal probability of choosing to move to C1 or C4. Given $(\mathbf{w}_{VGA}^{LS2-B5}, \gamma_{VGA}^{LS2-B5})$, there is an equal probability (0.487) of moving to C1 or C4. This aligns with our interpretation of Question 1: because the focal does not care about the corn kernels, it does not matter which way it goes when the banana is no longer available. However, this behaviour is also consistent with the corn having a positive reward since the choice of C1 or C4 leads to identical subsequent foraging (i.e., the focal receives corn at each step of its trajectory and the trajectories are equal length).

In summary, starting from different platforms does have an effect on the focal's behaviour, and our learned models support the hypothesis that the behaviour difference is driven by the proximity of B5 to the starting platform.

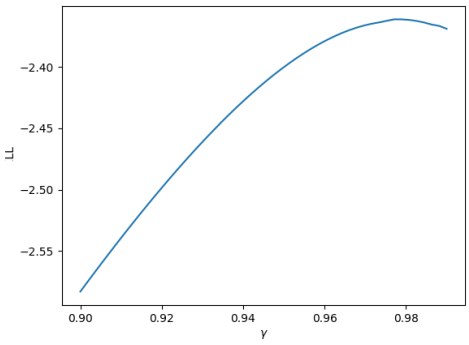
4.5.1 Convexity of the Objective Function w.r.t. the Discount Factor

The results suggest that, under the same computational budget, learning a policy for starting platform using vanilla gradient ascent method has the best overall performance. However, recall that in Chapter 3, we could not prove whether the objective function is concave or not w.r.t. $\mathbf{w}$ and γ jointly. More specifically, we only know that the objective function is concave w.r.t. $\mathbf{w}$; its convexity w.r.t. γ is inconclusive. Using gradient methods to solve a non-convex/non-concave problem can lead to getting stuck at a local optimum. Thus, to verify whether the solution obtained is a global optimum (or is close to the global optimum) w.r.t γ, we select $\mathbf{w}_{VGA}^{LS2-C1}$ and enumerate 50 values for γ from $[0.01, 0.99]$ and compute the objective function values, shown in Figure 4.5a. The objective function is a smooth curve that monotonically increases as γ increases until γ is very close to 1. Figure 4.5a also shows that the LL function is neither convex not concave w.r.t. γ, as we can see from the plot that it is convex from 0 to approximately 0.6, and becomes concave afterwards. If we zoom into the range of 0.9 to 0.99, as illustrated in Figure 4.5b, we can see that there exists a global optimum at around $\gamma = 0.98$. The objective function value using $\mathbf{w}_{VGA}^{LS2-C2}$ has a similar trend as

Figure 4.5a while using $\mathbf{w}_{VGA}^{LS2-B5}$ results in a function that monotonically increases with the best LL at $\gamma = 0.99$. These results empirically confirm our hypothesis made in Chapter 3 that the objective function is neither concave nor convex.



(a) LL by the discount factor $\gamma \in [0.01, 0.99]$.



(b) LL by the discount factor $\gamma \in [0.9, 0.99]$

Figure 4.5: The LL value by discount factor γ for starting from C1.

4.5.2 Alternative Modelling Choices

As mentioned in Section 4.3.1, to cast the foraging experiment as an MDP, we made a number of restrictive assumptions. These simplifications are due to the fact that we have limited information about the foraging experiments in the data. Here, we discuss several alternative modelling choices that can potentially lead to more realistic models that better represent the foraging experiment, assuming we have all the information needed in the given data. We hope that these suggestions can serve as a guide for future foraging experiment design and for what type of information would need to be collected for modelling.

4.5.2.1 Incorporating the First Decision

To incorporate the first decision into the MDP model, we can simply add a dummy state $s_d = (dummy, (1, 1, 1, 1, 1))$ to MDP-NoComp proposed in Section 4.3.1. We denote this MDP model as MDP-NoComp-SP, since the dummy node now serves as a fixed **s**tarting **p**latform. We assume that the distance between s_d and each platform is the same, denoted as d_0, and thus the immediate reward at this state is $R(s_d, a) = w_1\texttt{Food}(s_d, a) - w_2 d_0$. For illustration purpose, Figure 4.6 shows that the focal will start the trial in the middle of the pentagon, though in reality, the focal joins the experiment from outside the experiment site.

The advantage of having a dummy starting state, comparing to MDP-NoComp, is that it allows us to get a more complete picture of how the focal behaves and it aligns with the foraging experiments conducted in Arseneau-Robar et al. [10], where the focal's first decision was recorded. Thus, one may argue that MDP-NoComp-SP more closely adheres to reality and should be chosen as the mathematical representation for our application. However, a closer examination of MDP-NoComp-SP reveals that, given the ratio of corn to banana preference, the first decision will always favour going to B5. To see why this is the case mathematically, recall that the agent's optimal

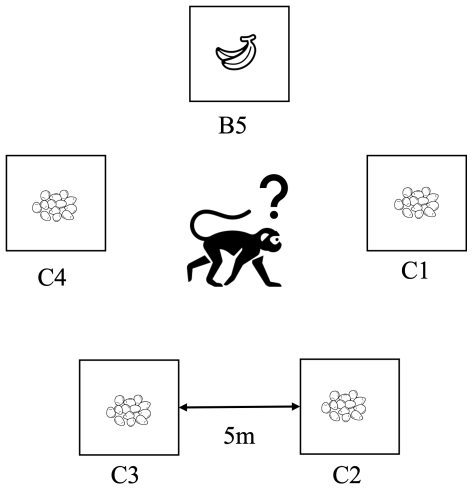


Figure 4.6: Illustration for adding a dummy state s_d to the MDP model.

policy simultaneously optimizes the value of every state in the MDP, $V(s), \forall s$. Assuming we have a deterministic policy, the optimal value of the dummy state s_d is

$$V^*(s_d) = [R(s_d, a_d) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \gamma^3 R(s_3, a_3) + \gamma^4 R(s_4, a_4)|\pi]$$

where the trajectory $\{(s_d, a_d), ..., (s_4, a_4)\}$ consists of the optimal actions at the states visited.

Given the structure of the reward function, we know that the path with the highest sum of reward is going around the array in order, no matter which platform the focal starts on. If the first choice is B5 (i.e., $a_d = B5$), the total discounted sum of reward of a path would be

$$R_{a_d=B5} = w_1 - w_2 d_0 + \sum_{t=1}^{4} \gamma^t (0.05 w_1 - 5 w_2)$$

If $a_d \neq B5$ and B5 is the j^{th} choice along the path, the discounted sum of reward becomes

$$R_{a_d \neq B5} = 0.05 w_1 - w_2 d_0 + \sum_{t=1}^{j-1} \gamma^t (0.05 w_1 - 5 w_2) + \gamma^j (w_1 - 5 w_2) + \sum_{t=j+1}^{4} \gamma^t (0.05 w_1 - 5 w_2)$$

If we compare the difference between these 2 paths

$$R_{a_d=B5} - R_{a_d \neq B5} = 0.95 w1 - 0.95 \gamma^j w_1 > 0$$

since $\gamma \in (0, 1]$ and we assume $w_1 > 0$.

Therefore, unless γ is 1, no matter what values w_1, w_2, and d_0 take, the resulting policy will always prioritize B5 as the focal's first choice. In the actual field data, we did not observe this type of behaviour: as mentioned in Section 4.4, only about 15% of the demonstrations started at B5, and the focal tends to choose one of the corn platforms as their first decision.

One potential cause of this misalignment between the model and the real behaviour is that, by including the dummy state in the MDP model, we assume that the first decision depends on the

same factors that affect the subsequent decision-making—food and distance. However, since the experiment site is set in an open area where the monkey may come from any direction, the first platform it chooses may depend on additional features such as which platform is closest to it, who is present in the audience, or the monkey's handling skill for getting the banana out of the container.

4.5.2.2 Choice of a Simple Combination of Features

Although the original study by Arseneau-Robar et al. [10] suggests that factors such as the focal's rank in the group also impacted its decision-making process, we did not include these elements in our reward function for the MDP.

Since our defined state space in the model is discretized based on the location of the focal, incorporating information such as age and rank is not feasible because they are dependent on the focal instead of the platforms. Thus, for the same state s, the "age" feature or the "rank" feature may have different values in two trajectories by different focals, resulting in an invalid environment for the MDP. One potential way to take these factors into account is to split the data according to the focal's status. Instead of simply splitting the data based on starting platforms, we can further divide each group into more granular subsets. For example, under the group in which the focals started at C1, a subset may be focals who were low-ranked adult monkeys in the group that started at C1. Further data exploration and testing is required to properly define these subgroups. One disadvantage of a more granular dataset is that we would have fewer data points in each set, making it challenging for the algorithm to learn parameters that are more generalized. However, it would be interesting to see how differently the monkeys perform the food-distance trade-off based on their status.

4.5.2.3 Alternative State Space

As we have seen in Section 4.3, the state space of our proposed model is small, with only 80 states in total. While a small state space enables us to compute certain elements that are usually infeasible to achieve (e.g., the enumeration of all possible paths), one disadvantage is that it may be too restricted to represent the actual foraging experiments realistically.

One of the most important simplifications we made to map the foraging experiment as an MDP is the assumption that the action of taking the food on a platform and travelling between platforms take no time. This simplification was made due to the fact that the original data only recorded the handling time of the focal getting the banana out, but not the other activities during the trial. Therefore, we do not see a suitable way to only incorporate handling time into the model.

Intuitively, we may think that the handling skill level can also affect which platform the focal decides to go to. Indeed, in Arseneau-Robar et al. [10], the authors show that the monkey's handling skill affects its first decision, which is not considered in our model. While it may not be obvious how being better or worse at getting the banana is relevant when foraging alone, when foraging in competition, the skill level may be critical to whether the focal can get the preferred food before its competitor does. One thing we can do is to process the timing of the handling into a "skill level" score for the focal. However, this again would result in a focal-based attribute rather than a location-based attribute as discussed in Section 4.5.2.2.

Assuming the data we are given contains sufficient and complete information, we can consider modelling the behaviour in a GridWorld setting, a popular framework for modelling RL and IRL

problems. Instead of specifying each state by platform, we can instead represent the state as a tuple of coordinates on the grid as well as the current time step similar to the model in Hirakawa et al. [49]. Each platform is located at certain coordinates and the focal can choose to move freely in the grid to collect the rewards.

One advantage of this potential model is its granularity: depending on the computational power and the recording device we have, we can adjust the grid size and record the time in shorter intervals to track the focal's movements. Further, we can also expand the definition of the experiment site to include area near the platforms. This way, we can track the focal's position before it reaches the first platform, thus incorporating the first decision into the model. Figure 4.7 shows the potential state space as a grid at a specific timestep. However, achieving this level of detail for the model requires

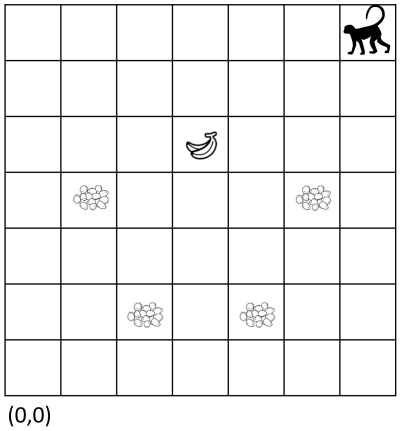


Figure 4.7: Example grid for the foraging experiment.

thorough planning before conducting the foraging experiments and devices that can precisely capture the position and the movement of the focal. One example is the experimental setup in the dataset used in Ashwood et al. [12], where a camera was placed under a maze to track mice behaviour.

4.5.3 Suggestions for Future Foraging Experiment Design

In summary, we present several alternative modelling choices for the foraging problem to create a more realistic representation under the restriction of modelling as an MDP. Since the field data we obtained for this study was not originally designed to be used under the IRL framework, we provide some suggestions for future foraging experiment design that can produce data more suitable for an IRL study.

- When recording the foraging behaviour, the data should include timing information.
- Implement an experimental setup that allows more precise movement tracking.
- In addition to the pentagon arrangement, we can also try increasing the number of platforms and setting up different configurations (e.g., a hexagon) to test generalizability of the learned discount factor and reward function.
- Following the previous suggestion, when we have more platforms, we can design more complex placements of food, such as having multiple bananas placed within the experiment site, or

having a third type of food as a reward. Such a set up can potentially elicit more diverse behaviour.

4.6 Conclusion

In this chapter, we used the modified MaxEnt IRL algorithm from Chapter 3 to examine the foraging behaviour of wild vervet monkeys when they forage alone. The original foraging experiment was conducted by Arseneau-Robar et al. [10], where the focus of the study was only the monkey's first and second decision in a trial. We first model the foraging experiment as an MDP, where the focal is the decision maker in the environment, and formulate the IRL problem whose unknown reward function is a linear combination of two features: food and travel distance. We then solve the IRL problem using the vanilla gradient ascent, iterative gradient ascent, and random search to recover the weight parameters and the discount factor. The results of our experiments indicate that vanilla gradient ascent exhibits the best overall performance. Moreover, the learned parameters suggest that, in the absence of competitive pressure, the monkeys exhibit farsighted decision-making behaviour by maximizing food reward while minimizing travel distance, aligning with the observations in Arseneau-Robar et al. [10].

While our simple model yields satisfactory results, it is important to acknowledge that, due to limited data availability, we had to make restrictive assumptions when representing the foraging experiment as a mathematical model. Therefore, we also discuss several alternative modelling options, such as a more complex reward function or a more realistic state space, provided the required data is accessible. These suggestions can guide future foraging experiment designs by specifying the type of information that should be collected for different modelling choices.

For the case of foraging alone, the mathematical results presented in this chapter match with our intuitions about the monkeys' behaviour. Our model provides explanations of the complete sequences of decisions made in a trial and can predict the monkey's actions given in any specific state. A particular novel hypothesis arising from our work is that the monkeys exhibit a future discounting factor close to 1, indicating under the sole foraging scenario, that they do little discounting of future reward. To our knowledge, this is the first attempt to analyse such as aspect of animal cognition based on behaviour data.

In the next chapter, we continue our study of wild vervet monkeys' behaviour and explore a more complicated case: foraging in competition, which presents additional challenges in formulating it as an IRL problem.

Chapter 5

Modelling Wild Vervet Monkey Behaviour When Foraging In Competition

5.1 Introduction

In this chapter, we continue our application of Inverse Reinforcement Learning (IRL) to animal behaviour. Extending the results from Chapter 4, we study the case when the focal monkey forages in competition. When modelling the case with no competitor, we treated the problem as a single-agent IRL problem since the focal is the sole decision maker. When foraging in competition, in addition to collecting the food reward, the focal may interact with its competitor(s). We define "focal" as the first monkey that reaches the experiment site while the "competitor" is defined as the monkey that arrives afterwards. Although it is possible for the focal to have more than one competitor, in the data it is less common to have three or more monkeys involved in a single trial. Thus, in this chapter, we consider the case where there is only one competitor and we ignored the records for cases where there was more than one (19.2% of the trials in the raw data).

Different from the case of foraging alone, we are interested in studying both the focal and the competitor behaviour during the trial, meaning the problem is now in a multi-agent setting. Recall in Section 2.5, we describe two ways of solving multi-agent IRL problems: casting a multi-agent problem to a single-agent problem by choosing to focus on one agent at a time and modelling the other agents as part of the environment, or modelling the problem as a Markov Game with two or more decision makers. In this chapter, we focus on the former method by embedding the competitor's action probabilities into the transition probabilities of the focal's MDP. This approach can be seen in Pinsler et al. [85], who investigated the behaviour of pigeons in a flock, where each pigeon was modelled independently.

We present two ways of extracting the competitor's behaviour. The first method is simple: we empirically calculate the action probabilities using the given demonstrations; the second method is more complex, where we adopt an iterative approach called level-k reasoning [30, 79], which assumes the monkeys are agents with a specific level of reasoning ability. We discuss these approaches in

Sections 5.2.2.2 and 5.2.2.3.

To our knowledge, this study is the first attempt to model monkey behaviour as a multi-agent IRL problem, and two research questions we are interested in are:

1. How different is the monkeys' behaviour when foraging in competition from when they forage alone?

2. How does the rank of the monkeys affect their decisions when foraging in competition?

This chapter is organized as follows. Section 5.2 introduces our methodologies for modelling the foraging experiment as an MDP. Next, experimental results are presented in Section 5.3, followed by a discussion about the limitations of our approaches and some alternatives we can explore in the future in Section 5.4. We then conclude this chapter in Section 5.5.

5.2 Methodologies

The experiment setup for foraging in competition is the exact same as described in Section 4.2. In addition to the assumptions we made for the foraging alone case, we present several further assumptions about the foraging experiments with a competitor:

1. Each trial begins when both monkeys are on a platform.

2. There is no interaction between the two monkeys during the trial. We assume the only actions that can be taken are to move to another platform or to stay.

3. Between the two monkeys, one is dominant over the other based on their ranks in the group. The focal can either be dominant or subordinate to its competitor.

4. If the dominant and the subordinate arrive at the same platform, the dominant would get the reward of the food and the subordinate would get 0.

Similar to assumptions we made for the foraging alone case, while the assumptions listed above are necessary for transforming the foraging experiment to an MDP, they also significantly simplify the experiment. Assumption 1 and 2, for example, make our model quite different from the real-world foraging experiment. Assumption 1 requires both the focal and the competitor to be on a platform to begin a trial since, as we have established in Section 4.3, we do not take the first decision of the monkeys into account when modelling their behaviour. Further, in the field experiment, it is possible for the competitor to join the trial after the focal has been to one or more platforms. However, when the competitor decides to join the experiment depends on factors that cannot be incorporated into the MDP model, such as how far away the competitor is from the experiment site when the focal starts foraging. Assumption 2 further simplifies the foraging experiment, since in reality, it is possible for the two monkeys to interact with each other. Since we do not have enough information to represent the interactions properly, these actions are omitted in the models.

Each monkey in the group has an assigned Elo rank [7, 33]; a higher Elo rank indicates the monkey is more dominant within the group. Assumption 3 is based on the fact that no two monkeys have the same rank. We call the monkey with a higher rank the "dominant" and the other "subordinate". Since the dominant has more power than the subordinate, we observe that the dominant is often

the monkey that gets the preferred banana reward while the subordinate tends to keep a distance from the dominant and rarely goes for the banana.

5.2.1 Modelling Foraging in Competition as an MDP

We denote the MDP model for foraging with one competitor as MDP-OneComp. Its state and action spaces are very similar to MDP-NoComp presented in Section 4.3.1, with slight modifications to incorporate the competitor into the state space. The transition probability, however, is no longer deterministic as in MDP-NoComp. Each known element of MDP-OneComp is described as follows.

- **State** S: we discretized the foraging experiment by platforms and each state consists of three elements:
  - `f_loc`: the platform the focal is currently on. `f_loc` $\in$ {C1, C2, C3, C4, B5}.
  - `remaining_plats`: a binary 5-tuple in which an element takes the value 1 if the platform has not been visited by the focal or the competitor. For example, if platform C1 is the only remaining platform, `remaining_plats` $= (1, 0, 0, 0, 0)$.
  - `c_loc`: the platform the competitor is currently on. `c_loc` $\in$ {C1, C2, C3, C4, B5}.

  The food is assumed to be taken when the focal/competitor visits a platform. Therefore, for each pair of `f_loc` and `c_loc`, if `f_loc` $\neq$ `c_loc`, there are 20 possible (`f_loc`, `c_loc`) pairs, and each pair has 8 possible combinations of `remaining_plats`; if `f_loc` $=$ `c_loc`, there are 16 possible combinations. In total, there are 240 states in the state space S. States with `remaining_plats` $= (0, 0, 0, 0, 0)$ are considered terminal states.
- **Action** A: an action $a \in$ {C1, C2, C3, C4, B5} represents the focal's choice of moving to another platform ($a \neq$ `f_loc`) or staying at the platform that it is currently on (i.e., $a =$ `f_loc`).
- **Transition Probability** P_{sa}: The transition probabilities are stochastic: transitioning into the next state $s' = (\texttt{f\_loc}', \texttt{remaining\_plats}', \texttt{c\_loc}')$ not only depends on the focal's action a but also the competitor's. Since the competitor is now part of the environment, we assume that its action distribution is known beforehand. For example, for $s = (\text{C1}, (0, 0, 1, 1, 1), \text{C2})$, if $a =$ B5 and we know that at C2, the competitor has a 50% probability of going to C3, then the transition probability to $s' = (\text{B5}, (0, 1, 1, 1, 0), \text{C3})$ is 0.5.

The structure of the reward function of MDP-OneComp is assumed to be the same as MDP-NoComp: a linear combination of weighted food and distance features. Namely,

$$R(s, a) = w_1 \texttt{Food}(s, a) - w_2 \texttt{Distance}(s, a).$$

However, there is a subtle difference between a dominant focal and a subordinate focal. In Assumption 4, we state that the dominant monkey has the priority of access to the food reward if both monkeys arrive at the same platform. Therefore, the dominant's reward function is defined as:

$$R^{\text{Dom}}(s, a) = \begin{cases} w_1^{\text{Dom}} \texttt{Food}(s, a) - w_2^{\text{Dom}} \texttt{Distance}(s, a) & \text{if going to a platform with food} \\ -w_2^{\text{Dom}} \texttt{Distance}(s, a) & \text{if going to a platform with no food} \\ 0 & \text{if staying at the current platform} \end{cases} \tag{5.1}$$

And the reward function for a subordinate focal is:

$$R^{\text{Sub}}(s,a) = \begin{cases} w_1^{\text{Sub}}\texttt{Food}(s,a) - w_2^{\text{Sub}}\texttt{Distance}(s,a) & \text{if going to a platform with food} \\ -w_2^{\text{Sub}}\texttt{Distance}(s,a) & \text{if going to a platform with no food or the same platform as the dominant} \\ 0 & \text{if staying at the current platform} \end{cases} \tag{5.2}$$

Note that, different from the reward function in Section 4.3.1, we do not penalize the monkey for staying at the same platform. When foraging in competition, we notice that the monkeys sometimes choose to stay and observe its competitor's action. Further, while we assume that both the dominant and subordinate are optimizing the trade-off between food and travel distance, the weights for the feature values may be different. Similarly, the monkeys may also value the future reward differently based on their ranks. Therefore, our goal is to learn two sets of parameters: $(\mathbf{w}^{\text{Dom}*}, \gamma^{\text{Dom}*})$ and $(\mathbf{w}^{\text{Sub}*}, \gamma^{\text{Sub}*})$ for the dominant and the subordinate focal, respectively.

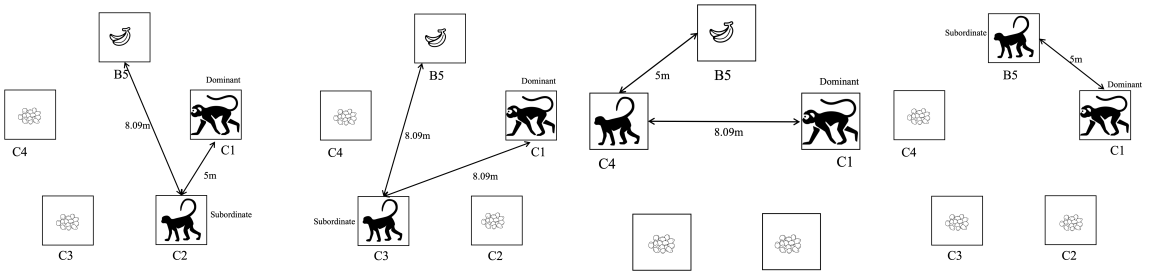
Finally, since we assume that the experiment only starts when both monkeys are on a platform, there are 10 possible initial states as shown in Table 5.1. Different from the case of foraging alone which only has three starting states (Figure 4.2 in Section 4.3.1), the 10 starting states are determined by the two factors combined: the distance between dominant (either as the focal or the competitor) and subordinate and the distance between the subordinate and platform B5. Therefore, states such as (C1, $(0,0,1,1,1)$, C2) and (C2, $(0,0,1,1,1)$, C1) are not symmetrical because although the distances between the two monkeys are the same (5m), the distances between the subordinate and B5 are different: C2 to B5 is 8.09m, and C1 to B5 is 5m. The four of the ten possible starting states where the dominant is at C1 are shown in Figure 5.1.

Table 5.1: Ten potential starting states depending on the monkeys' locations.

Dominant Location	Subordinate Location	Starting State s
C1	C2	(C1, $(0,0,1,1,1)$, C2)
	C3	(C1, $(0,1,0,1,1)$, C3)
	C4	(C1, $(0,1,1,0,1)$, C4)
	B5	(C1, $(0,1,1,1,0)$, B5)
C2	C1	(C2, $(0,0,1,1,1)$, C1)
	C3	(C2, $(1,0,0,1,1)$, C3)
	C4	(C2, $(1,0,1,0,1)$, C4)
	B5	(C2, $(1,0,1,1,0)$, B5)
B5	C1	(B5, $(0,1,1,1,0)$, C1)
	C2	(B5, $(1,0,1,1,0)$, C2)

With all the components for the MDP described above, the key of constructing MDP-OneComp is how we define the transition probabilities. Recall that we assume that the focal can either be the dominant or the subordinate to its competitor (Assumption 3). When the focal is dominant, the transition probabilities reflect the action probabilities of a subordinate, and vice versa.

In the next section, we present two approaches for extracting the competitor's action distributions. The first one empirically extracts the competitor's action distributions using the given demonstrations; the second is an iterative approach called level-k reasoning [30, 79] from behavioural game theory.



(a) Focal starting at C1 and competitor at C2.

(b) Focal starting at C1 and competitor at C3.

(c) Focal starting at C1 and competitor at C4.

(d) Focal starting at C1 and competitor at B5.

Figure 5.1: Four possible starting states with focal at C1.

5.2.2 Methods for Extracting the Competitor's Action Probabilities

5.2.2.1 Data Processing

Since the first approach relies on the given data to determine the transition probabilities, we first discuss the data processing procedure for the case of foraging in competition, which is significantly more complicated than the data processing step described in Section 4.4.

When there is no competitor involved, the data transformation process is simple. We only need the order of platforms visited by the focal, which is readily available in the dataset. However, for the case with a competitor, the data is much more complex as we now have to extract the behaviour of both the focal and the competitor. Three example records from the original dataset are shown in Table 5.2.

Table 5.2: Two example trials recorded using ethograms.

	Focal	Competitor
Example 1	C3 C4 C2	C1 B5
Example 2	C3 rsC4.rsC2	ae-n.paC1.paB5
Example 3	C3 ap-ag.vo C4.B5	ae-n.rsC1 rt rsC2

Besides the platform codes (i.e., C1 to B5) we have seen before, there are additional annotation such as "rs" or "pa" before the platforms as shown in Example 2. The platform codes and the text are called ethograms, which are commonly used to record the behaviours of animals in ethology [107]. Each ethogram (e.g., C1 or rs) represents an observed behaviour exhibited by the study species, and when put together, we get a sequence of actions a monkey took in a foraging trial.

The space and the periods between two ethograms hold different meanings in the sequence: the space between the ethograms is a "while" relationship, and the "." indicates a "then" relationship, that the monkey performed the actions in a consecutive manner regardless of the actions of the other monkey.

In Example 1, the focal visited platform C3, C4 and C2, and the competitor visited C1 and B5. With the spaces between the ethograms, we can describe Example 1 in plain language as:

> The trial starts with the focal on C3 and competitor on C1. Then, while the focal moved to C4, the competitor moved to B5 simultaneously. Finally, the focal moved to C2 while the competitor stayed at B5.

Notice, at the end of this example, we assume that the competitor stayed at B5 because there is no information about what the competitor did when the focal moved to C2. To transform the data into trajectories that we can use, we pad a monkey's incomplete sequence by the last action it took. Therefore, a complete trajectory for Example 1 is: {(C3, $(0, 1, 0, 1, 1)$, C1), (C4, $(0, 1, 0, 0, 0)$, B5), (C2, $(0, 0, 0, 0, 0)$, B5)}.

In Example 2, the raw data becomes slightly more complicated with additional ethograms, such as "ae-n" which represents the monkey **a**pproaches the **e**xperiment **n**eutrally. That is, the monkey joins the trial without displaying any hostile or aggressive tendencies towards its competitor. Since we only focus on which platforms each monkey went to when modelling its behaviour mathematically, we ignore the non-platform-related actions (i.e., "rs", "ae-n", and "pa") when creating a trajectory. Therefore, Example 2 in plain words would be:

> While the focal was on C3, the competitor was on C1 then it went to B5. While the competitor is on B5, the focal went to C4 then C2.

Similar to how we encode the end of Example 1, we make the assumption that a monkey simply stayed at the last platform it was on when we do not have enough information. Thus, a trajectory for Example 2 would be: {(C3, $(0, 1, 0, 1, 1)$, C1), (C3, $(0, 1, 0, 1, 1)$, B5), (C4, $(0, 1, 0, 0, 0)$, B5), (C2, $(0, 0, 0, 0, 0)$, B5)}.

Finally, the most restrictive assumption we made is Assumption 2 in Section 5.2, in which we assume that there is no interaction between the two monkeys in the trial. However, Example 3 illustrates the possible interactions between the monkeys, often in the form of the dominant aggressing the subordinate. Let us focus on the ethograms "ap-ag.vo" for the focal and "rt" for the competitor. The former reads as "approached aggressively then aggressively vocalized" at the competitor, and the latter means the competitor "retreated", meaning it moved away from the focal to avoid the aggression. When processing trials with interactions, we simply remove the interaction ethograms along with other non-platform ethograms. Thus, Example 3 becomes: "C3 C4.B5" and "C1 C2", which can then be translated into a feasible trajectory similar to Example 1.

5.2.2.2 Empirical Model

The first model directly uses the available training data to calculate the empirical probabilities of the competitor moving from one platform to another. These probabilities are then embedded into the transition probabilities P_{sa}. Consider an example of a simplified foraging experiment shown in Figure 5.2.

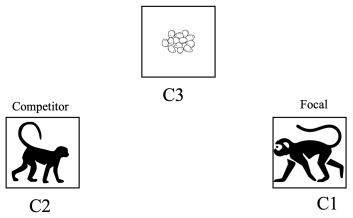


Figure 5.2: A simplified foraging experiment with three platforms.

In this example, we only have three platforms: C1, C2, and C3, and a starting state: $s_0 =$ (C1, $(0, 0, 1)$, C2), and the competitor can either move to C1 or C3, or stay at the current platform. In

this empirical model, we extract the competitor's action probabilities by counting the number of transitions observed between pairs of platforms in the given data, and normalize the frequencies to obtain the probabilities. For example, assume that the demonstration set for the foraging experiment in Figure 5.2 only consists of three trajectories: $\xi' = \{\tau_1, \tau_2, \tau_3\}$ where

$$\begin{aligned}
\tau_1 &= \{(\text{C1},\ (0,0,1),\ \text{C2}),\ (\text{C3},\ (0,0,0),\ \text{C3})\} \\
\tau_2 &= \{(\text{C1},\ (0,0,1),\ \text{C2}),\ (\text{C3},\ (0,0,0),\ \text{C3})\} \\
\tau_3 &= \{(\text{C1},\ (0,0,1),\ \text{C2}),\ (\text{C3},\ (0,0,0),\ \text{C2})\}.
\end{aligned}$$

Based on ξ', when at C2, the competitor stayed at C2 once, moved to C3 two times, and never moved to C1. Thus, the competitor's action distribution is:

$$\begin{aligned}
P_{comp}(\text{C2 to C2}) &= \frac{1}{2+1} = 0.333 \\
P_{comp}(\text{C2 to C3}) &= \frac{2}{2+1} = 0.667 \\
P_{comp}(\text{C2 to C1}) &= 0.
\end{aligned}$$

The MDP for this example would then have a transition probability that includes P_{comp}. The transition probability at s_0 is shown in Table 5.3.

Table 5.3: Transition probability at s_0 = (C1, (0, 0, 1), C2) for the example in Figure 5.2 with empirical action probabilities of the competitor.

State s	Action a	State s'	Transition Probability $P_{sa}(s')$
(C1, (0, 0, 1), C2)	C1	(C1, (0, 0, 1), C2)	0.333
		(C1, (0, 0, 0), C3)	0.667
	C2	(C2, (0, 0, 1), C2)	0.333
		(C2, (0, 0, 0), C3)	0.667
	C3	(C3, (0, 0, 0), C2)	0.333
		(C3, (0, 0, 0), C3)	0.667

When the agent takes an action a in state s, the resulting state s' is probabilistically generated based on the transition probabilities in Table 5.3. With all the known components of the MDP, we can then formulate an IRL problem to recover the agent's reward function parameter $\mathbf{w}$ and the discount factor γ.

For the actual foraging experiment, we denote the MDP model that uses the empirical transition probability as MDP-OneComp-E-D for dominant as focal, and OneComp-E-S for subordinate as focal. We obtained the transition probabilities in the same way as the example presented above: by normalizing the frequencies of platform pairs in the demonstrations. If the dominant is the focal, the transition probabilities depend on the behaviour of the subordinate, and vice versa. Table 5.4 and 5.5 illustrate the action probability of a dominant and a subordinate, respectively.

From Table 5.4, we can see that a dominant monkey has the highest probability of moving towards B5 when at C1 (0.5), and is more likely to move to an adjacent corn platform when at C2, C3, or C4. From Table 5.5, the subordinate tends to either stay or simply move to the adjacent corn platform, and it almost never moves to B5. In the rare case where the subordinate reaches B5,

Table 5.4: A dominant's empirical transition probability.

To From	C1	C2	C3	C4	B5
C1	0.053	0.158	0.211	0.079	0.5
C2	0.311	0.067	0.444	0.044	0.133
C3	0.069	0.103	0.069	0.759	0
C4	0.3	0.05	0.45	0	0.2
B5	0.083	0.019	0.037	0.176	0.685

Table 5.5: A subordinate's empirical transition probability.

To From	C1	C2	C3	C4	B5
C1	0.565	0.217	0.13	0.043	0.043
C2	0.157	0.333	0.451	0.039	0.02
C3	0.073	0.091	0.218	0.6	0.018
C4	0.184	0.053	0.132	0.632	0
B5	0.04	0	0	0.1	0.86

it behaves in the same way as a dominant at B5: a high probability of staying (0.685 for dominant and 0.86 for subordinate) rather than going to another platform.

It should be noted that the probabilities shown in the tables assume the platform that the competitor moves to is still available. For example, in Table 5.4, a dominant at C1 would move to C2 with a probability 0.158 if and only if all four platforms are available. Thus, if certain platforms no longer have the food reward, we assume that the competitor would not visit those platforms again and we renormalize the probabilities.

Consider a simple example where we have a subordinate focal at the state s_t= (C3, $(1,0,0,0,1)$, C2), that is, the subordinate is currently at C3 and the dominant at C2, with platform C1 and B5 remaining. According to Table 5.4, the dominant would have 0.311 probability of going to C1, 0.067 of staying at C2, and 0.133 of going to B5. Since there are only two remaining platforms to go to at this state, we renormalize the probabilities of going to either one of those or staying: $P(\text{C1}|s_t) = \frac{0.311}{0.311+0.067+0.133} = 0.609$, $P(\text{C2}|s_t) = \frac{0.067}{0.311+0.067+0.133} = 0.131$, and $P(\text{B5}|s_t) = \frac{0.113}{0.311+0.067+0.133} = 0.26$.

One disadvantage of using empirical data to calculate the transition probability is that, if the dataset is too small, certain transitions may not be available. Although using the data we have, Table 5.4 and 5.5 account for all possible transitions in the state space, there exists a corner case where we have 0 transition probability to any state. Consider a subordinate focal at state s'_t= (C3, $(0,1,0,0,0)$, C4); if the dominant has 0 probability of staying or moving to C2, s'_t becomes an absorbing state. To handle this corner case, we manually reassign a 50% chance of staying at the current platform, and the remaining 50% is evenly distributed to the available remaining platforms. Therefore, in this example, assuming the action $a' = \text{C2}$, there is a 0.5 probability of transitioning to s'_{t+1} =(C2, $(0,0,0,0,0)$, C2) or a 0.5 probability to (C2,$(0,0,0,0,0)$, C4).

5.2.2.3 Iterative Model With Level-k Reasoning

One of the pitfalls of the previous model is that it heavily depends on data availability. We can see from Table 5.4 and 5.5 that there are several transition probabilities that take the value of 0 because

the corresponding actions were not observed in the data. Given sufficient data, we may believe that it is generally true that a dominant would never choose to go to C4 from C3 even if there is food on C4, as per Table 5.4. However, since foraging in competition is far less common than foraging alone, the data for foraging trials with two monkeys is limited. Therefore, in our case, it seems to be more reasonable to believe that the dominant never visits C4 from C3 because of data limitation. While the empirical model has several advantages, we seek to move away from relying on the data to compute the action probabilities while trying to represent the assumption that the focal has some idea of what its competitor will do.

In this method, the transition probabilities reflect the competitor's strategy at any give state. That is, we treat the competitor as a decision maker and obtain its optimal policy, which is then embedded into the MDP with the focal as the agent in the environment. Let us revisit the three-platform example in Figure 5.2. Instead of counting the number of times the competitor visited C2 and C3 to calculate P_{comp} as described in Section 5.2.2.2, let us assume that we already have a policy of the competitor beforehand, $\pi_{comp}(a|s)$. To obtain a policy for the competitor means that we would treat the competitor as the "focal" and the original focal as the "competitor" and formulate an MDP where the competitor is the decision maker. Thus, the original starting state $s_0 =$ (C1, $(0,0,1)$, C2) becomes $s_0' =$(C2, $(0,0,1)$, C1). Assume the competitor's policy at s_0' is

$$\pi_{comp}(\text{C1}|(\text{C2, } (0,0,1), \text{ C1})) = 0.01$$
$$\pi_{comp}(\text{C2}|(\text{C2, } (0,0,1), \text{ C1})) = 0.09$$
$$\pi_{comp}(\text{C3}|(\text{C2, } (0,0,1), \text{ C1})) = 0.9,$$

the transition probabilities for the focal at the original s_0 are shown in Table 5.6.

Table 5.6: Transition probability at $s_0 =$ (C1, $(0,0,1)$, C2) for the example in Figure 5.2 with competitor policy π_{comp}.

State s	Action a	State s'	Transition Probability $P_{sa}(s')$
(C1, $(0,0,1)$, C2)	C1	(C1, $(0,0,1)$, C1)	0.01
		(C1, $(0,0,1)$, C2)	0.09
		(C1, $(0,0,0)$, C3)	0.9
	C2	(C2, $(0,0,1)$, C1)	0.01
		(C2, $(0,0,1)$, C2)	0.09
		(C2, $(0,0,0)$, C3)	0.9
	C3	(C3, $(0,0,0)$, C1)	0.01
		(C3, $(0,0,0)$, C2)	0.09
		(C3, $(0,0,0)$, C3)	0.9

Note that s_0 and s_0' are symmetrical and so we can always convert a state in the "competitor as focal" MDP back to a state in the original state space, and map the competitor's policy to the transition probabilities accordingly.

In this method, to acquire a policy of the competitor to be embedded into the focal's MDP, we need the focal's policy to construct the competitor's MDP. To achieve this iterative process, we borrow a concept of iterative learning from behavioural game theory [24], called quantal level-k model [30, 79], where π_{comp} is assumed to be from a competitor that reasons at a lower level than the focal.

Level-k Reasoning

Level-k reasoning is based on the idea that human decision makers are not perfectly rational and, rather, have bounded rationality [99], meaning that we have limited cognitive resources and can only perform a bounded number of recursions of strategic reasoning. When in a game, each player can reason strategically about what the others may do before choosing his/her own actions. It is reasonable to believe that, with humans, we can take into account the strategic reasoning abilities of another again: an agent i takes an action by reasoning that agent j reasons about i. Such a recursive reasoning can continue (i.e., agent i reasons that agent j reasons that i reasons about j) until the cognitive limit k.

The level-k model associates an agent with an intelligence level $k \in \{0, 1, 2, ..\}$, which corresponds to the number of reasoning recursions this agent can perform:

- Level-0 agents do not have the capability to perform any strategic reasoning
- Level-k ($k \geq 1$) agents make strategic decisions by treating other agents as level-$(k-1)$ agents.

A simple example that more concretely illustrates the concept is the guessing games experiment conducted by Nagel [79]. In the experimental setup, there are 1000 people in the experiment, each of them needs to pick a number of from 0 to 100. The person who is closest to half of the average gets a reward. A level-0 player would choose a number randomly and uniformly from the range, while a level-1 player would choose 25 by assuming everyone else is level-0 and so the mean of all the other choices will be 50. The player would get more and more rational as the intelligence level k increases, and so the number the player chooses gets closer and closer to 0 as it attributes higher rationality to other players. Zero is the Nash equilibrium of the guessing game.[1]

Naive Level-k Model

In reality, most humans are level-2 players [102]. In our application, we assume that monkeys are also intelligent agents who take their competitor's actions into account. Therefore, we propose an iterative modelling approach that represents the focal as a level-k decision maker. We denote each MDP model as MDP-OneComp-k-D for a level-k Dominant and MDP-OneComp-k-S for a level-k Subordinate.

The naive level-k learning framework is shown in Figure 5.3. Given an initial level-0 policy, $\pi^{0,Dom}$ and $\pi^{0,Sub}$, and a set of demonstrations with a dominant/subordinate focal (i.e., ξ^{Dom} and ξ^{Sub}), we iteratively create level-k dominant/subordinate by embedding the level-$(k-1)$ subordinate/dominant policy into the transition probabilities.

We refer to the level-0 policy as the "anchoring policy", which is generated either randomly or with an educated guess. In our study, we choose to define the anchoring policy for both dominant and subordinate as follows. At each state s, we assume that 90% of the time, the monkey would move to an available platform or stay at its current platform, while 10% of the time it would make an irrational decision of moving to a visited platform. One may suggest that we can also use the empirical policies presented in Section 5.2.2.2 as the anchoring policies. However, note that the action probabilities extracted from the given data likely already incorporate the monkeys' foraging strategies with the knowledge of their competitor (e.g., the subordinate rarely visits B5 in the

[1] In Nagel's experiment, the mean is 27.05 and the median is 17 [79], suggesting that the subjects are at level-2.

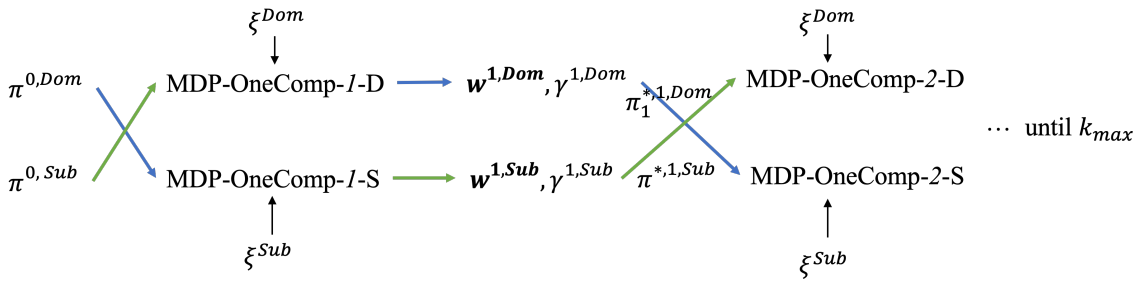


Figure 5.3: The iterative naive level-k framework.

presence of the dominant). Using such a policy as an anchoring policy conflicts with the assumption that a level-0 agent does not have the ability to reason strategically.

At each level, we take the same set of demonstrations and learn a pair of $(\mathbf{w}^{k,i}, \gamma^{k,i})$ and subsequently a policy $\pi^{*,k,i}$ using the opponent's previous level's policy $\pi^{*,k-1,-i}$ ($\{i,-i\} = \{Dom, Sub\}$) as part of the transition probabilities.

As in the empirical model, we embed the other monkey's actions as transitions into the MDP model. However, instead of using the global empirical probabilities, the action distributions come from a level-$(k-1)$ policy. The advantage of the level-k approach is that, compared to the empirical models, we no longer directly depend on the demonstrations to determine the transition probabilities. Instead, we assume that the focal has the capability to reason about its competitor's potential actions at any given state by assuming the competitor is one level below itself. As a result, we obtain a more granular action distribution compared to the ones in Section 5.2.2.2. In MDP-OneComp-E, we have a fixed probability of moving to a platform regardless of where the focal is, whereas in the level-k model, the action probabilities are state-specific, which take the focal's position into account. However, modelling using this iterative approach is more computationally demanding than the empirical models: to reach the models for a level-k Dominant and level-k Subordinate, we need to create and solve $2k$ MDPs in total.

5.2.3 Computing the Gradients of the Objective Function

By embedding the competitor's action probabilities into the environment, the transition dynamics of the MDPs become stochastic. Thus, for every feasible trajectory $\tau = \{(s_0, a_0), ..., (s_{|\tau|}, a_{|\tau|})\}$, we define a probability $q(\tau)$ that is induced by the MDP's transition probabilities $P(s'|s,a)$ alone:

$$q(\tau) = \mathcal{I}(s_0) \prod_{t=1}^{|\tau|-1} P(s_{t+1}|s_t, a_t).$$

In the original MaxEnt IRL framework proposed by Ziebart et al. [129], the authors state that the likelihood of a trajectory τ can be approximated using

$$P(\tau|\mathbf{w}, \gamma, P_{sa}(\cdot)) = \frac{q(\tau)e^{U(\tau)}}{Z_p} \tag{5.3}$$

where $Z_p = \sum_\tau q(\tau)e^{U(\tau)}$ is the partition function.

Under the MaxEnt IRL framework, the objective function (Eq. 3.8 in Section 3.2) becomes

$$\arg\max_{\mathbf{w},\gamma} \sum_{\tau\in\xi} \log P(\tau|\mathbf{w},\gamma,P_{sa}(\cdot)) = \frac{1}{|\xi|}\sum_{\tau\in\xi} U(\tau) - \log Z_q + \frac{1}{|\xi|}\sum_{\tau\in\xi}\log q(\tau). \tag{5.4}$$

The last term, $\frac{1}{|\xi|}\sum_{\tau\in\xi}\log q(\tau)$, in Eq. 5.4 is a constant that does not depend on $\mathbf{w}$ or γ. Therefore, the gradients w.r.t. $\mathbf{w}$ or γ of Eq. 5.4 is equivalent to Eq. 3.9 and 3.10. Namely,

$$\nabla_{\mathbf{w}} = \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t) - \sum_{\tau} P(\tau)\sum_{t=0}^{|\tau|}\gamma^t\phi(s_t)$$

$$\nabla_{\gamma} = \frac{1}{|\xi|}\sum_{\tau\in\xi}\sum_{t=1}^{|\tau|} t\gamma^{t-1}\mathbf{w}^\top\phi(s_t) - \sum_{\tau} P(\tau)\sum_{t=1}^{|\tau|} t\gamma^{t-1}\mathbf{w}^\top\phi(s_t)$$

Recall, in Section 3.3, we stated that using Algorithm 4, the second terms of the gradients can be computed by using time-indexed state visitation frequencies (Eq. 3.11 and 3.12). However, as noted in Snoswell et al. [100], using Eq. 3.11 and 3.12 for stochastic MDPs leads to "approximate gradients and negatively impacts the reward learning process". The authors provided empirical evidence in their computational experiments and showed that the gradients computed based on state visitation frequencies resulted in greater learning error compared to other approaches tested [100]. Although the proof for the claim about gradient approximation is currently not available in Snoswell et al. [100], we also observed the same negative effect when conducting preliminary experiments. When using Eq. 3.11 and 3.12, the gradient ascent method could sometimes take a step in the wrong direction, hindering the learning process.

Thus, for the case of foraging in competition, instead of using Algorithm 4 as we did for the deterministic MDP in Chapter 4, we compute the gradients using Eq. 3.9 and 3.10. This is, of course, generally intractable and computationally expensive as we need to enumerate all feasible trajectories with length $|\tau|$ in every iteration to compute the partition function Z_q. However, given that the state and action spaces in our application are manageable, we chose this enumeration approach to enable more accurate learning.

In general, the approach of using state visitation frequency for gradient computation based on Ziebart et al. [129] works well for deterministic MDPs (as we have shown in Chapter 4), but the same cannot be said if we have stochastic transition dynamics. Further investigation is required to study the cause of this issue.

5.3 Experiments

5.3.1 Data

For the case of foraging with one competitor, originally there are 299 trials in the dataset. After the data processing procedure described in Section 5.2.2.1, we filter out trajectories that did not start with both monkeys being on a platform and incomplete trajectories with length of 1, and 126 trajectories remain. Of these 126 trajectories, 73 have a dominant focal and 53 have a subordinate focal. To increase the number of trajectories for each rank, we switch the `f_loc` and `c_loc` in each state in the trajectories to create a mirrored dataset for the other rank. Therefore, each rank has

126 trajectories in total and we obtained the training and test set through a random 80-20 split. there are 93 trajectories in the training set and 33 in the test set.

5.3.2 Experimental Setup

For the computational experiments, similar to the experimental setup in Section 4.2, we randomly and uniformly initialize the parameters as follows:

$$w_1 \in U[10, 11]$$
$$w_2 \in U(0, 0.1]$$
$$\gamma \in U[0.01, 0.99]$$

The ranges of w_1 and w_2's initialization ensure that the reward of going to an adjacent platform will be non-negative while going to a non-adjacent platform will result in a negative reward except for going to the banana platform. Additionally, we set the temperature $\sigma = 1$. Since the experiments in Chapter 4 show that we obtain relatively good results within approximately 30 iterations (Figure 4.3), we set the maximum number of iterations for each model to `max_iter` $= 30$. The learning rate $\alpha_{\mathbf{w}}$ and α_γ are set to [0.02, 0.01] and 0.05 for all models.

Finally, we adopt the vanilla gradient ascent (VGA) method during training since VGA produces the best overall results in Chapter 4.

5.3.3 Results

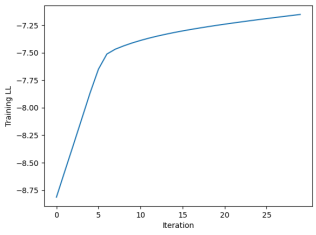
5.3.3.1 Training Results

Table 5.7 illustrates the learned parameters for the different models. Overall, the weight parameters are similar across different methods but, for the discount factor γ, we can see that while the dominant focal is a more farsighted decision maker with higher value of γ, the subordinate is more myopic with lower γ. Interestingly, for both dominant and subordinate, the value of γ increases with level k. In particular, the discount factor for the level-2 dominant is 0.99, suggesting that the agent is very farsighted.

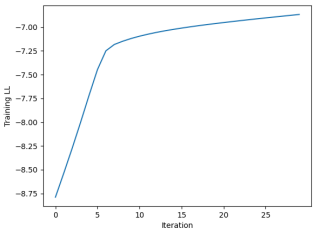
Table 5.7: Learned parameters by models.

Model	Parameters		
	w_1	w_2	γ
Empirical Dominant	10.242	0.311	0.817
Level-1 Dominant	10.24	0.335	0.855
Level-2 Dominant	10.238	0.336	0.99
Empirical Subordinate	10.098	0.688	0.575
Level-1 Subordinate	10.095	0.69	0.576
Level-2 Subordinate	10.101	0.65	0.654

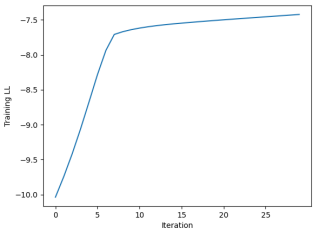
Figure 5.4 shows the best training LL at each iteration for different models. Figure 5.4a to 5.4c show the results for a dominant focal, and Figure 5.4d to 5.4f show the results for a subordinate focal. We can see that while the best LLs plateaued quickly for models with a dominant focal, the subordinate models improved steadily over the iterations.



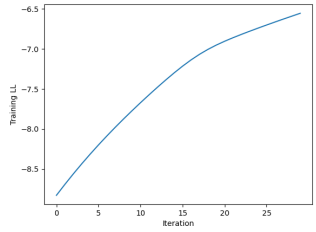
(a) Training LL for Empirical Dominant focal.



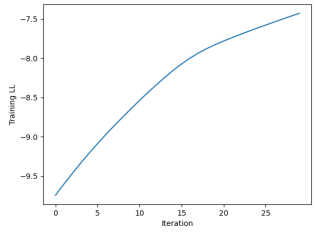
(b) Training LL for Level-1 Dominant focal.



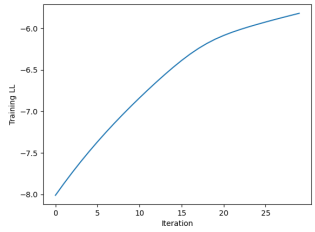
(c) Training LL for Level-2 Dominant focal.



(d) Training LL for Empirical Subordinate focal.



(e) Training LL for Level-1 Subordinate focal.



(f) Training for Level-2 Subordinate focal.

Figure 5.4: Best training LL in each iteration by model.

5.3.3.2 Test Results

Table 5.8 shows the experimental results on the test set. We use the test set log likelihood (LL) as the main metric for performance and level-1 dominant and level-2 subordinate achieve the best LL among the models by rank, respectively. For the test feature match, while the difference between expected travel distance in the test set and from the learned policy is relatively small, the food feature difference is more significant considering that corn has a value of 0.05 and banana of 1.

Table 5.8: Test results of different models by ranks and levels.

Model	Test LL	Feature Difference – Food	Feature Difference – Travel Distance
Empirical Dominant	-7.179	-0.688	2.095
Level-1 Dominant	-6.878	-0.69	1.913
Level-2 Dominant	-7.339	-0.695	2.014
Empirical Subordinate	-7.187	-0.869	0.623
Level-1 Subordinate	-8.004	-0.877	0.499
Level-2 Subordinate	-6.387	-0.833	0.101

Interestingly, the combination of large negative food feature difference and small travel distance difference suggests that our learned parameters lead to the prioritization of banana for both dominant and subordinate. Recall that the feature difference is calculated by subtracting the learned expected feature counts from the demonstrations' feature count. Therefore, the learner travels roughly the same distance as the monkeys in the demonstration (the maximum expected travel distance difference is around 2.1 meters, less than the distance between two adjacent platforms), but obtains more food in a trial. Given the feature values assigned to corn (0.05) and banana (1), the most probable

explanation is that the learner would choose to move to B5 more often than the demonstrated data.

5.4 Discussion

In this chapter, we extend our application of wild vervet monkey foraging behaviour presented in Chapter 4 to the case where the focal forages with a competitor. Based on the experiment results, we can try to answer the research questions we posed at the beginning of this chapter.

1. How different is the monkeys' behaviour when foraging in competition from when they forage alone?
In terms of the trade-off between the two different factors in the reward function—food and travel distance—we can see that the focal monkey values the food and travel distance similarly to the case of foraging alone. More specifically, although we deliberately initialize the weight parameters w_1 and w_2 to ensure a positive reward for going to the adjacent platform, the learned w_2 is significantly larger than the initial value, which is less than 0.1. Thus, as in the foraging alone case, the reward function indicates that only moving to the banana platform B5 leads to positive reward while all other actions have a non-positive reward. Consider the level-1 dominant, for example, whose learned weight parameters are $\mathbf{w}^{1,Dom} = (10.242, 0.311)$. If B5 is the adjacent platform, the focal will get a reward of $10.242 \cdot 1 - 0.311 \cdot 5 = 8.687$; if B5 is non-adjacent, the reward is $10.242 \cdot 1 - 0.311 \cdot 8.09 = 7.72$. For any corn platform, the focal will get -1.043 if the platform is adjacent and -2 if it is non-adjacent. Thus, similar to what we observed for foraging alone, there is no incentive for the focal, regardless of its rank, to get the corn; the monkeys only care about the banana.

For the discount factor, we know that the focal is a farsighted decision maker with γ greater than 0.95 when foraging alone. As shown in Table 5.7, regardless of the focal's rank, all models learn a smaller discount factor value (except for the level-2 dominant focal). Comparing to foraging alone, we may interpret the lower γ values as that the focal values the future reward less when in competition. Intuitively, this interpretation is reasonable since the future is more uncertain as the competitor is also foraging in the trial. When the focal is alone, a high discount factor implies that it knows that all rewards are available to itself only. However, a competitor represents a potential threat to the focal that the future reward may be taken, prompting the focal to focus more on the present. We can see that a dominant focal is still a farsighted decision maker with $\gamma > 0.8$, while the subordinate focal is more myopic. We will discuss more about the difference between dominant and subordinate in the next question.

2. How does the rank of the monkeys affect their decisions in the trial?
As alluded to in the previous discussion, the main difference between a dominant and subordinate focal is how they value the future reward compared to the present. The weight parameters for both focals are quite similar (see Table 5.7), indicating that the monkeys have a similar trade-off between food and travel distance regardless of their ranks. This is a reassuring result as we intended the food weights to reflect the intrinsic value of the food that should be independent of focal rank.

As level-1 dominant and level-2 subordinate are the models with the best test LL (see Table 5.8), our results suggest that a subordinate focal is more myopic than the dominant ($\gamma = 0.654$ vs. 0.855) and it uses a higher level of reasoning k. One possible explanation for the subordinate's lower

discount factor value is that, since the subordinate understands that it is at a disadvantage when competing with the dominant, it focuses more on the immediate reward rather than planning for the future. Furthermore, the different values of k suggests that the dominant monkey does not need to reason as much as its competitor since it has more power; while the subordinate, on the contrary, needs to perform more complex strategic reasoning to maximize its own reward.

From the experiment results, we observe that while the dominant prioritizes the banana reward, the subordinate tends to stay on the current platform and rarely moves to B5. Consider an example where the dominant is at platform C1 and the subordinate at C2, and C3, C4, and B5 are available. The best models, level-1 dominant and level-2 subordinate, have a 0.95 and 0.01 probability of moving towards B5, respectively. The level-2 subordinate is most likely to stay at C2 with 0.89 probability. While it is obvious why a dominant focal would move almost deterministically to B5 and a subordinate would rarely go to B5, it may seem counter-intuitive for the subordinate to stay at C2 instead of moving to an available platform. Based on our domain knowledge and observations from the actual foraging experiments, we know that the subordinate has a fair probability of moving to an adjacent platform (see Table 5.5). Thus, there exists a misalignment between our expectations and learned behaviour from the IRL problem.

After the banana has been taken, the focal tends to stay at its current platform even if there are still available platforms in the trial since, as discussed in Section 4.5, the MDP models do not have the action "leave experiment". For a dominant focal, the most probable action at any state where B5 is no longer available is to stay with probability around 0.65; for a subordinate focal, the probability of staying is approximately 0.9. This result suggests that, in a foraging trial, the most probable actions for a dominant agent would be to take the banana and stay at B5, while a subordinate agent would simply stay at its current platform from the start, and neither of the two agents would bother moving to the corn platforms.

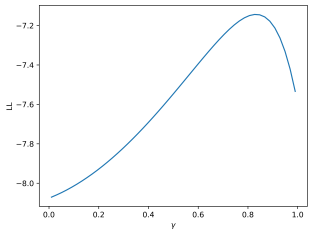
This discrepancy between the model prediction and real-life behaviour points to one of the limitations we have in the modelling process. The tendency to stay at the current platform is because we did not assign a penalty for staying at the current platform. Based on the behaviour in the data, we observe that, unlike the foraging alone case, a monkey may stay at its current platform for different reasons, such as observing the competitor's next move before making a decision. Further, after the data processing, the trajectories are often padded with the action of staying. Consider the case where the focal is at B5, one example trajectory from the dataset is {(B5, $(0,1,1,1,0)$, C1), (B5, $(0,0,1,1,0)$, C2), (B5, $(0,0,0,1,0)$, C3), (B5, $(0,0,0,0,0)$, C4}. In this example, the focal was at B5 while its competitor took all the remaining corn. Recall that the banana was placed inside a container and thus, depending on the monkey's handling skill, the time it takes for the monkey to get it out of the container may vary. Since we discretize the foraging trials by each decision made by the monkeys, this example trajectory is to be interpreted as the focal chose to stay three times while in reality, a single decision was made and it took a while to finish the task. By not imposing a negative reward on staying, we accommodate this action seen in the processed trajectories, but we also misrepresent handling time with staying.

It is also unclear whether adding the action "leave experiment" to the MDP models would yield different results for the foraging in competition case. Regardless of the value we assign to "leave" (positive, 0, or negative), it has effectively the same impact as staying in terms of matching the food and distance features as both feature values would be 0.

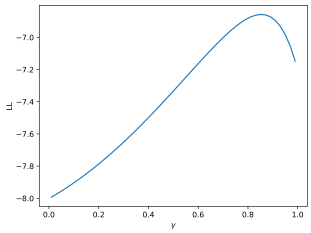
Moreover, although we initially set staying to be an unfavourable action (moving to an adjacent corn platform would yield a positive reward), the learned parameters make it a more attractive choice than moving to a non-banana platform, which has a negative reward. For a subordinate monkey, staying at its current platform is in fact the optimal choice. Since the dominant can take the reward if the two monkeys move to the same platform as the dominant, the subordinate will receive a negative reward for travelling (i.e., $-w_2$`Distance`). Thus, there is no real incentive of taking any action in a trial except for staying. In reality, however, this result is a limitation of our models and a consequence of the restrictive data processing procedure, and we should be cautious about how we interpret the results for the case of foraging in competition.

5.4.1 Convexity of the Objective Function w.r.t. the Discount Factor

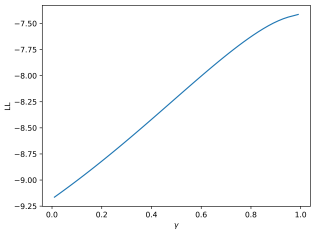
Recall in Chapter 4, we plotted the objective function values by γ to empirically show that the objective function is neither concave nor convex w.r.t. the discount factor. Since the number of trajectories in the case of foraging in competition is still manageable to enumerate, we can compute the log likelihood of each model with the learned weight parameters and 50 γ values from $[0.01, 0.99]$ and see if there exists any local optimum w.r.t. γ. Figure 5.5 shows the objective function for each model vs the discount factor γ. Figure 5.5a to 5.5c show the LL of the empirical dominant



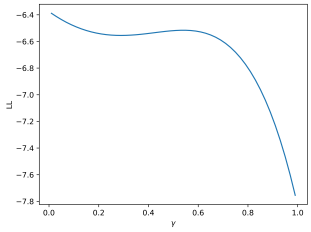
(a) LL for Empirical Dominant focal by γ.



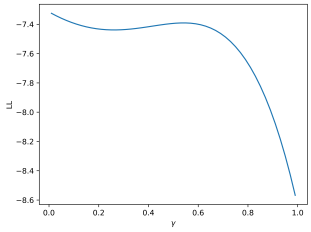
(b) LL for Level-1 Dominant focal by γ.



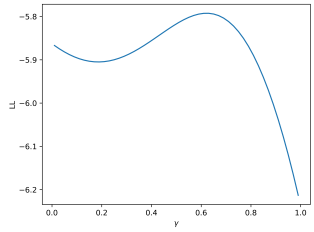
(c) LL for Level-2 Dominant focal by γ.



(d) LL for Empirical Subordinate focal by γ.



(e) LL for Level-1 Subordinate focal by γ.



(f) LL for Level-2 Subordinate focal by γ.

Figure 5.5: LL for each model by γ.

focal, level-1 and level-2 dominant focal, respectively, and Figure 5.5d to 5.5f show the subordinate models in the same order. We can see that for dominant focal, the trend is quite similar to the case of foraging alone (Figure 4.5) where the LL is monotonically increasing until the global optimum, or in the case of Level-2 dominant, the optimal solution is when γ is at the border 0.99. For the subordinate focal, however, the LL behaves in a different way. For the empirical and level-1 subordinate models, there exists a global optimum at $\gamma = 0.01$, while the level-2 subordinate model

has a global optimum at around $\gamma = 0.65$. Thus, depending on the initial γ value, vanilla gradient ascent may lead to different optimal discount factor values for the subordinate models. Interestingly, we can see that the level-1 models (Figure 5.5b and 5.5e) are more aligned with the models with empirical competitor distributions (Figure 5.5a and 5.5d) in terms of the shape of the objective function, while the level-2 models are quite different from the others.

Further, the global optimum in Figure 5.5c and 5.5f suggest the level-2 focals are farsighted. It is unclear why the level-2 models, both for dominant and subordinate, indicate the focals would be more farsighted than the level-1 models. We only know that a higher value of k indicates that the decision maker is more rational, but being rational is not equivalent to being farsighted. One hypothesis is related to the temperature parameter σ in the soft VI [130] algorithm. Recall that σ represents how rational the agent is in soft VI, and we set σ to be 1 for all models. One potential issue about the level-k models then may be as we increase k, it is possible that σ should decrease to indicate the fact that rationality is increasing. Further investigation is required to study the relationship between k and σ.

5.4.2 Alternative Modelling Approaches

5.4.2.1 A Markov Game with Bounded Rationality

In the two models presented in this chapter, instead of treating both monkeys in the foraging trial as agents in the IRL problem, we embed the competitor into the environment of the MDP and represent its action through the transition probabilities. One natural extension would be to include the competitor as a decision maker and model the foraging experiment as a Markov game (MG).

Here, we briefly discuss one possible framework based on the Bounded Risk-sensitive Markov Game (BRSMG) proposed by Tian et al. [106]. In fact, the authors also utilize level-k reasoning and thus render the MG into a single-agent IRL problem, making the model very similar to the naive level-k model we presented in Section 5.2.2.3. In the MG formulation, the level-$(k-1)$ competitor's policy is represented explicitly in the MG, while in our MDP model, it is embedded into the transition dynamics. Additionally, in BRSMG, the decision makers are assumed to be risk-sensitive and the agent's intelligence level k is an unknown parameter. In our application, to make BRSMG equivalent to our level-k model, we would relax these two conditions by assuming the monkeys are risk-neutral and their intelligence levels are bounded by k. We can then formulate our problem as a classical MG as presented below. Interested readers are referred to Tian et al. [106] to learn more about the BRSMG framework.

Recall that a finite classical MG is a tuple $\langle \mathcal{P}, S, A, P_{sa}, R, \gamma \rangle$, where P is the set of agents in the game. In our case, we have two agents in the game, $\mathcal{P} = \{Dom, Sub\}$. $S = (\texttt{d\_loc}, \texttt{remaining\_plats}, \texttt{s\_loc})$ and $A = A^{Dom} \times A^{Sub}$ are the joint state and action spaces. Note that S is a combination of the location of dominant and subordinate, $\texttt{d_loc}, \texttt{s_loc} \in \{$C1, C2, C3, C4, B5$\}$, and the corresponding configurations of the remaining platforms. The action set A is $A^{Dom} = A^{Sub} = \{$C1, C2, C3, C4, B5$\}$. The reward $R = (R^{Dom}, R^{Sub})$ where $R^i(s, a^i, a^{-i})$ represents the immediate reward of agent i in state s based on its own and the other player's action ($-i = P \backslash \{i\}$). Similar to the reward function for the MDP models described before, $R^i(s, a^i, a^{-i}) = \mathbf{w}^i \phi^i(s, a^i, a^{-i})$, where $\mathbf{w}^i$ and $\phi^i(s, a^i, a^{-i})$ are the weight parameters and the feature value for agent i, respectively. Different from our naive level-k model, the transition probability $P_{sa} = S \times A \to S$ is now deterministic given

the dominant's and subordinate's actions at any state. Finally, $\gamma = (\gamma^{Dom}, \gamma^{Sub})$ is the discount factor for each agent.

A policy of agent i is denoted as π^i and, since we assume the decision maker is risk-neutral, the optimal policy $\pi^{*,i}$ maximizes the expected sum of discounted rewards with respect to its opponent $-i$'s policy.

$$\pi^{*,i} = \text{argmax}_{\pi^i} V^{\pi^i}(s), \quad \forall s \in S \tag{5.5}$$

where

$$V^{\pi^i}(s) = E_{\pi^{-i}}[R(s, a^i, a^{-i}) + \gamma^i V^{\pi^i}(s')]. \tag{5.6}$$

When solving an MG, the goal is to find the Markov perfect equilibrium (MPE) when each agent's policy is optimal given the policies of the other players, which is NP-hard [106]. However, recall that we are incorporating the concept of bounded rationality into the model and so we assume that for an agent i with intelligence level k, its optimal policy is with respect to a level-$(k-1)$ opponent $-i$. Therefore, the value function of a level-k agent i is:

$$V^{\pi^{i,k}}(s) = E_{\pi^{-i,k-1}}[R(s, a^i, a^{-i}) + \gamma^{i,k} V^{\pi^{i,k}}(s')]. \tag{5.7}$$

Similar to the naive level-k model, we can then iteratively obtain the optimal policy of one agent at level k and use it for the other at level-$(k+1)$, effectively reducing the MG to an MDP. In Tian et al. [106], the authors also formulate the IRL problem under the MaxEnt framework. Let us denote $\hat{p} = (\mathbf{w}^{Dom}, \mathbf{w}^{Sub}, \gamma^{Dom}, \gamma^{Sub})$ to consolidate the parameters to be learned for notation clarity and, following Tian et al. [106], define the objective function as:

$$\max_{\hat{p}} \sum_{\tau \in \xi} \log P(\tau|\hat{p}) = \sum_{\tau \in \xi} \log \Pi_{t=0}^{|\tau|-1} P(\hat{a}_t|s_t, \hat{p}), \tag{5.8}$$

where $P(\hat{a}_t|s_t, \hat{p})$ is the joint probability of agents' actions at state s_t [106]. Namely,

$$P(\hat{a}_t|s_t, \hat{p}) = \pi^{*,i,k^i}(s_t, a_t^i)\pi^{*,-i,k^{-i}}(s_t, a_t^{-i}). \tag{5.9}$$

Notice that, although the MG formulation has deterministic transition probabilities, when reducing the MG to an MDP, the transition dynamics become stochastic since they depend on the action distributions of the level-$(k-1)$ competitor. Therefore, as discussed in Section 5.2.3, computing the gradients of the objective function using Algorithm 4 as we did in Chapter 3 may lead to recovering a set of inaccurate parameters $\hat{p}$.

The main difference between this modified BRSMG framework and our naive level-k model is the competitor is seen as a decision maker in the game rather than being part of the transition dynamics of the model. However, since both approaches assume the monkeys have bounded intelligence level and that one monkey always makes its decisions based on the assumption that the other is one level below itself, we can expect that using the modified BRSMG framework will result in similar results compared to the naive level-k model.

5.4.2.2 Alternative State Space

A More Descriptive State Representation for More Granular Action Probabilities

While the two methods presented in this chapter produce a good approximation of the competitor's behaviour, both provide a rather high-level representation of the action distributions. As mentioned previously, transition probabilities in the naive level-k models are more granular than the empirical model because the focal's position is taken into account. However, anecdotally, we learned[2] that there may be more factors that affect the competitor's actions than where the focal is.

One potential factor is how many platforms the competitor has visited and taken the food from. This factor is particularly applicable to a subordinate competitor, who, based on observations from field experiments, would stop foraging after visiting a certain number of platforms even if there is still food left. This behaviour is interpreted to mean that the subordinate respects the hierarchy, and by letting the dominant take the food, the subordinate is less likely to be aggressed or displaced. Thus, in our current model, for the same state representation (e.g., (C1, $(0, 0, 0, 1, 1)$, C2)), the competitor's action probabilities may differ based on whether C2 is the first or second platform it visited.

Another factor that affects the competitor's decision-making may be how full it is at the beginning of the trial. This factor applies to both a dominant and subordinate competitor. Since the foraging experiment site is set in an open area, the same monkey can visit the site and join the experiments multiple times in a short period of time. Moreover, we can also infer that the feeding platforms are not the monkeys' only food source. Intuitively, we would expect that if the monkey is full at the beginning, it may be less motivated to move compared to if it is hungry.

We can expand the state representation to include these two factors as non-negative integers, which we denote as `f_food_num` and `f_fullness` for the focal and `c_food_num` and `c_fullness` for the competitor. Thus, the new state becomes a 7-tuple (`f_loc`, `f_food_num`, `f_fullness`, `remaining_plats`, `c_loc`, `c_food_num`, `c_fullness`), and each transition increases the new factors' values by 1. For the initial state s_0, `food_num` would be set to 0 but determining the initial fullness of the monkeys is challenging. Although we can keep track of how many times the same monkeys has done the foraging experiment, activities outside the experiment are not recorded, thus, we do not have any information on whether the monkeys have other food sources.

Incorporating Interactions Into the State Space

One of the greatest simplifications we made is to assume that both the focal and the competitor focus solely on maximizing their food reward and minimizing the travel distance. In reality, interactions between the monkeys are often observed in the foraging experiments (e.g., Example 3 in Table 5.2) and they are part of the social context that is crucial for understanding the monkeys' foraging behaviour.

Following the current 3-tuple structure of a state, one way to incorporate interactions into the model is to define specific `f_loc` and `c_loc` that represent the interactions while `remaining_plats` is unchanged. In Example 3 in Table 5.2, "ap-ag" (approach aggressively) and "vo" (aggressive vocalization) can be seen as two types of aggression from the focal and "rt" (retreat) is the competitor's reaction. Thus, in addition to the platforms C1 to B5, we can expand the value `f_loc` and `c_loc` can take by defining a set of interactions $I = \{\text{ap-ag, vo, rt, ...}\}$. That is, $\texttt{f\_loc}, \texttt{c\_loc} \in \{\text{C1}, ..., \text{B5}\} \cup I$.

[2] T. Jean M. Arseneau-Robar, personal communication.

Subsequently, we need to expand the action space (i.e., $A \in \{\text{C1}, ..., \text{B5}\} \cup I$) and define the transition probabilities and the immediate reward for the new states and actions. One example transition is described as follows. Assume we have a dominant focal and the current state is $s = (\text{C1}, (0, 0, 0, 1, 1), \text{C2})$) and the focal chooses action $a =$ ap-ag. We can define a deterministic transition probability of 1 to $s' = (\text{ap-ag}, (0, 0, 0, 1, 1), \text{rt})$, meaning that the subordinate competitor will be forced to retreat instead of going to another platform or staying. Conversely, if, at the same state $s = (\text{C1}, (0, 0, 0, 1, 1), \text{C2}))$, we have a subordinate focal and the dominant competitor chooses to attack, we would transition into $s'' = (\text{rt}, (0, 0, 0, 1, 1), \text{ap-ag})$ deterministically, regardless of what the action is.

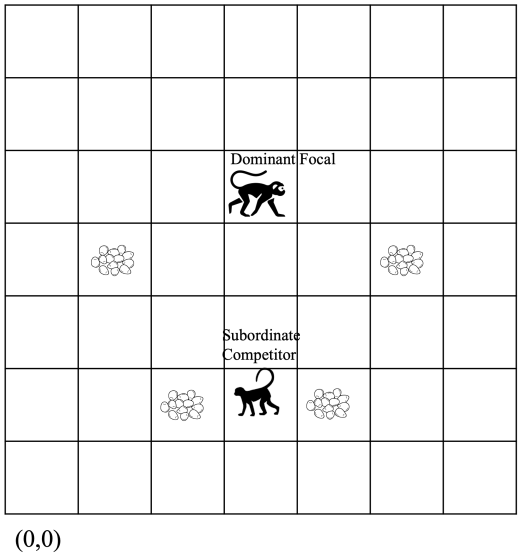
Finally, the most challenging part of expanding the state space to handle interaction is defining the reward function value for these states. The main reason why we decided not to include interactions in our current model is because we do not have enough information to quantify the reward for aggressing the opponent and for retreating from an attack. We hypothesize that the monkey may be driven by intrinsic motivations, such as asserting dominance over the opponent, or by how far away it is from the opponent. Further studies are required to identify the features that are relevant to the interactions.

GridWorld as the Environment

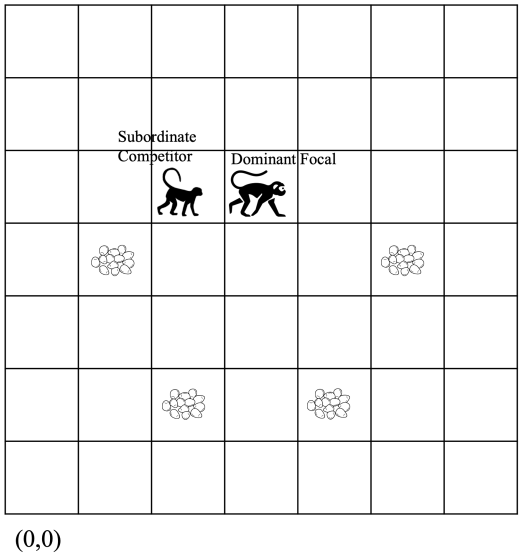
Another alternative model would be to represent the foraging experiment in a GridWorld setting with time steps. As described in Section 4.5.2, each state becomes a tuple of coordinates on the grid and the current time step. The advantages of this approach mentioned in Section 4.5.2 also apply to the case of foraging in competition, including more accurate tracking of the monkeys' movements and their positions before reaching the first platform, which can be used to incorporate the first decisions into the model. An additional benefit of the grid setting is that it allows us to measure the distance between the focal and the competitor. In our current state representation, the distance between the monkeys is fixed to be either 5 or 8.09 metres depending on the platforms they are on. In a grid setting, however, we can calculate the relative distance more accurately. For example, Figure 5.6 shows two different states at an arbitrary time step t. We can see that as the grid size increases, the distance between the two monkeys becomes more accurate. Knowing the relative distance between the focal and the competitor can help verify the hypothesis about when interactions occur during the trial. Intuitively, we would assume that the focal is more likely to interact with the competitor in Figure 5.6b than Figure 5.6a since the subordinate is much closer. We can then incorporate the relative distance as a feature in each state and in the reward function.

5.4.2.3 Incorporating the Relationship Between Focal and Competitor

We also simplify the social relationship between the focal and the competitor in our model. Since the monkeys live together as a group, we can expect that, outside the foraging experiment, they would have social interactions that either strengthen or weaken their relationship. One example is social grooming: an activity where one monkey cleans or maintains the other's body or appearance to create social bonds or in exchange for favours [95]. We can hypothesize that the relationship between a specific focal-competitor pair will affect their action probabilities during the trial. For example, the dominant may allow the subordinate to collect more food reward and may be less likely to attack the subordinate if they are "friends". While the original data recorded some interactions



(a) Possible state with the focal at $(3, 4)$ and the competitor at $(3, 1)$.



(b) Possible state with the focal at $(3, 4)$ and the competitor at $(2, 4)$.

Figure 5.6: Two possible states in a grid setting.

between trials, we did not find it sufficient to quantify the relationships between any two individuals.

We currently only have two classes of monkeys—dominant and subordinate—which is determined by the monkeys' Elo ranks. One limitation of this classification is that we cannot differentiate trials where the monkeys have similar Elo ranks from those with drastically different ranks. We may expect that the interactions between two monkeys may differ based on how similar their Elo ranks are. For example, consider two monkeys with relatively high ranks in the group in the same trial. They may both be considered "dominant" in the group but based on our current classification method, one will be classified as the subordinate and so is expected to behave as a subordinate. However, this high-ranked "subordinate" may be more likely to challenge the other monkey and fight for the banana.

Similar to the discussion of incorporating age and sex of each monkey into the model in Section 4.5.2, one way we can include the social relationships among the monkeys is to further split the dataset based on these specific conditions. However, such a split requires a significant amount of data to ensure that each subset would have sufficient data for training.

5.4.3 Suggestions for Future Foraging Experiment Design

In summary, to create more realistic mathematical representations of the foraging experiments, we propose the following suggestions on the design of future experiment that can lead to better data availability and suitability for an IRL study.

- One of the biggest issues for studying the case of foraging in competition is that we have very limited data. From the field data, it is obvious that foraging in competition is less common than foraging alone since the monkeys can choose if they want to participate in the trial and when to join the trial. Thus, perhaps a more controlled environment should be considered for

collecting data for foraging in competition. By conducting controlled trials, we can deliberately select focal-competitor pairs to gain more precise insights into the monkeys' behaviour.

- Increase the number of platforms in the experiment. Currently, if we assume both focal and competitor would always choose to move to an available platform, each trial would end in 3 steps as we only have 5 platforms. If we want to capture more diverse behaviour and study the monkeys' decision sequences more in depth, we should increase the number of platforms available. With a larger experiment site, we can then consider the case where there are more than two monkeys involved in a trial. The social dynamics among the monkeys may elicit behaviour that is not observed in the current data. For example, if there are two subordinates and one dominant in a trial, we may see the two subordinates cooperating against the dominant while maximizing their own rewards.

- Similar to the case of foraging alone, implementing precise movement tracking and including timing information when collecting the data will allow us to build more sophisticated mathematical models that better represent the real-life foraging experiment.

5.5 Conclusion

In this chapter, we extended the foraging experiment presented in Chapter 4 to study the foraging behaviour of wild vervet monkeys in competition. We focus on the case when there is only one competitor in the trial, and transform this multi-agent problem back to the single-agent setting by embedding the competitor's behaviour in the transition probabilities of the MDP model. We present two methods of obtaining the action distributions of the competitor: directly extracting from the given data, or by adopting the level-k reasoning framework, which iteratively produces level-k policies that can be used as the transition probabilities of level-$(k+1)$ models.

We were also interested in studying how a monkey's rank affects their decisions in a foraging trial. We formulate the MDP models from two perspectives: a dominant monkey as the focal with a subordinate competitor and a subordinate focal with a dominant competitor, and aim to recover the focals' weight parameters and discount factors. We solve the IRL problem using the vanilla gradient ascent and the experimental results suggest that when foraging in competition, the dominant tends to be a farsighted decision maker and the subordinate tends to more myopic.

While we believe that our models provide novel insights on wild vervet monkeys' behaviour when foraging in competition, it is important to note that, similar to modelling for foraging alone in Chapter 4, we made restrictive assumptions when transforming the foraging experiment into a mathematical model due to limited data availability. To enable better designed IRL studies on foraging behaviour in the future, we present several alternative modelling options, such as incorporating interactions between dominant and subordinate during the trial into the model. We also provide some suggestions on future foraging experiment designs that may lead to more realistic models.

In conclusion, for the case of foraging in competition, the experimental results presented in this chapter show some interesting differences compared to the results in Chapter 4. We hope the exploratory efforts presented in this chapter can serve as a starting point for future research on animal behaviour using IRL, particularly in the multi-agent setting.

Chapter 6

Conclusion

6.1 Summary and Contributions

In this thesis, we explored the framework of inverse reinforcement learning (IRL) with unknown discount factor, building on Giwa and Lee [40, 41], and applied it to study animal behaviour. Currently, the majority of IRL studies focus solely on recovering the reward function and treat the discount factor as a fixed parameter. However, we believe that intelligent decision makers perceive the future differently from one and another. Indeed, our experiments on wild vervet monkeys' foraging behaviour show that they discount the future reward differently based on the scenario they are in (foraging alone or in competition) and their social status (being a dominant or a subordinate monkey).

In Chapter 3, we examined the framework for solving an IRL problem with an unknown discount factor proposed by Giwa and Lee [40, 41]. We noticed several minor errors in the proposed formulations. In particular, we found that the given the structure of the gradients (Eq. 3.11 and 3.12), the visitation frequency matrix should be indexed by time (Algorithm 4). We then investigated a claim about the convexity of the objective function w.r.t. the weight parameters and the discount factor jointly. We attempted to provide a proof for this claim and demonstrated that, without knowledge of the given demonstrations, it is not trivial to arrive at such a conclusion. We also discussed the connection between different formulations of the objective function seen in the literature and derived the conditions required to make them equivalent to each other.

In Chapter 4, we applied the method presented in Chapter 3 to an application of animal foraging behaviour. In this chapter, we focused on wild vervet monkeys' behaviour when they forage alone. We first modelled this foraging problem as a finite deterministic Markov decision process (MDP), with the focal monkey as the decision maker and the reward function assumed to represent a trade-off between the food reward and the travel distance. The recovered parameters from our experiment suggest that when foraging alone, the monkeys prioritize the preferred food reward (banana) and they are very farsighted decision makers with a high discount factor. This result provides novel insight into monkeys' decision-making process and offers explanations for the observed behaviour. We also discussed several alternative modelling options that may lead to MDPs that can more accurately represent the real-life foraging experiment. Finally, using the results we obtained, we proposed suggestions for future foraging experiment designs that are more oriented to an IRL study.

In Chapter 5, we extended the findings from Chapter 4 to the case when the focal monkey forages with a competitor. Different from the case of foraging alone, both the focal and the competitor are decision makers in the trial, and we cast this multi-agent problem to a single-agent setting by embedding one of the two monkeys' actions into the environment. We proposed two methods of extracting the competitor's behaviour, either through empirically calculating the action probabilities using the given demonstrations or through an iterative approach called level-k reasoning [30, 79]. Since the transition probabilities of the MDPs are stochastic, we found that the algorithm proposed in Chapter 3 could not be used. Thus, we chose to compute the required gradients through enumeration instead. Our results suggest that, when foraging in competition, the monkeys with lower rank (i.e., the subordinate monkey) need to perform more complex strategic reasoning than the dominant and they are generally more myopic. Although this insight is interesting and points towards a research direction exploring the relationship between power dynamics and foraging behaviour in animals, we acknowledge that our models for the foraging in competition case are prototypes due to data limitations. Therefore, we should be cautious about how we interpret the experimental results. As in Chapter 4, we presented an extensive discussion on alternative models and provided suggestions on future experiment design.

6.2 Future Work

Alternative Modelling Options in Chapter 4 and 5

We proposed various alternative modelling choices in Section 4.5.2 for the case of foraging alone and in Section 5.4.2 for foraging in competition.

The first suggestion is to incorporate the first decisions made by the focal and the competitor into the models. As we have discussed in Section 4.5.2, if we assume that the first decision depends on the same factors that affect the subsequent decision-making—food and travel distance—the model would always choose to move to B5 first. Since the experiment site is set in an open area where the monkey may approach from any direction, the first platform it chooses could be influenced by additional factors, such as which platform is closest to it or who is present in the audience. Further, in the case of foraging in competition, the competitor may only choose to join the experiment after the focal monkey has started foraging and visited multiple platforms. Therefore, when modelling the competitor, we need to identify not only factors that affect which platform it may choose first but also when it chooses to join the experiment. The latter is difficult within our current modelling framework because it is possible that, when the focal first started foraging, the competitor was approaching the site and was simply too far away to join at that moment. Since we do not have data about the competitor's distance from the site, predicting when it would join is very challenging.

The second suggestion involves implementing movement tracking and including timing information when collecting the behavioural data. With accurate movement and position tracking, we can model the foraging experiment in a GridWorld setting where the states are defined not by platforms but as tuples of coordinate and timestep. This future research direction can lead to a significantly more realistic model for the foraging problem. By including timesteps in the state space, we can incorporate the banana handling time into the model; and with a finer grid, we can accurately calculate the distance between the two monkeys, which can potentially provide insights into when interactions occur during the trial (e.g., if the two are close to each other).

The third suggestion is to split the data into different subsets based on the characteristics of the monkeys. For the foraging alone case, we can explore how, or if, the monkeys perform the food-distance trade-off differently based on their status (e.g., male vs. female, adult vs. subadult). For the foraging in competition case, splitting the dataset by the relationship between different focal-competitor pairs could also yield interesting insights.

However, as noted in Section 4.5.2 and 5.4.2, all the above suggestions require a substantial amount of data. Thus, if we want to implement these changes in the future, we will need to rerun the foraging experiments to collect more data.

Feasible Next Steps for MDP-OneComp Using the Current Data

Some immediate next steps we can take using the available data we currently have are described as follows.

1. Adding "attempt to get food" and "leave experiment" to the model.

Recall that in Section 4.3, Assumption 3 states that once a monkey visits a platform, the food is assumed to be taken. When foraging alone, this statement is true since all the rewards are available to the focal monkey only, thus, it does not need to "try" to get the food. However, when foraging in competition, it is often observed that the subordinate attempts to get the food on a certain platform but fails due to aggression from the dominant. This behaviour can be seen as having additional uncertainty when moving to another platform in the transition probabilities. Further, in Section 4.5, we state that one of the potential causes of why the focal would continue foraging even with a negative utility for corn is that the MDP models do not allow the monkey to leave the experiment.

Adding "attempt to get food" and "leave experiment" to the model would lead to a more realistic representation of the foraging experiment. Consider the following changes we need to make to incorporate the notions of "attempting" and "leaving".

- Action space A. The action set now consists of six actions: $a \in \{$C1, C2, C3, C4, B5, leave$\}$. Each action a (except for "leave") also represents "attempting" implicitly with a certain probability of transitioning to a state where the food is not taken.

- State space S. In our current state space, the `remaining_plats` always reflect the fact that the food has been taken at the focal and the competitor's location by setting the corresponding element in the tuple to be 0. For example, if the focal is at C1 and the competitor at C2, the first two elements of `remaining_plats` are 0 (e.g, $(0, 0, 1, 1, 1)$ or $(0, 0, 0, 1, 1)$). To incorporate "attempting" into the model, we can have an additional state, s_{fail}, that represents the monkey fails to get the food. If the focal is at C1 and takes action "C3" while the competitor stays at C2, then one of the possible states it transitions to may be s_{fail} = (`failed`, $(0, 0, 1, 1, 1)$, C2), where `failed` is an arbitrary location that indicates the focal fails to get the corn on C3. Incorporating "leaving" into the model is simpler. If the focal chooses to leave the experiment, then the trial ends and we transition into a terminal state s_{exit}. If the competitor chooses to leave while the focal stays in the trial, then we set `c_loc` = `exit`, and the next state solely depends on the focal's action a.

- Transition probabilities P_{sa}. If we assume that the action "attempting" can sometimes lead to success, then the transition probability of taking an action a does not only depend on the competitor's action probabilities but also the success rate of trying. If the focal is at C1 and

takes action "C3", we assume that there is a probability p of success (i.e., $s' =$ (C3, $(0,0,0,1,1)$, `c_loc`)) and $1-p$ of attempting but failing (i.e., $s' =$ (`failed`, $(0,0,1,1,1)$, `c_loc`)), where `c_loc` is the location of the competitor. The transition probabilities associated with "leaving" are deterministic: if the focal chooses to leave, we transition into s_{exit} with a probability of 1; if the competitor chooses to leave, the model then becomes equivalent to one for the foraging alone case. For example, if we are currently at $s =$ (C1, $(0,0,0,1,1)$, `exit`) and $a =$ B5, we would transition to (B5, $(0,0,0,1,0)$, `exit`) deterministically.

For "attempting to get food", note that since we do not add any extra state to represent the aggression that leads to a failed attempt, the interaction is implicitly embedded into the transition probability. Namely, there is a probability of $1-p$ of getting aggressed by the dominant competitor.

The immediate reward of moving to another platform still depends on the value of the food and travel distance, as the current model. However, we now assume that with probability $(1-p)$, the focal would try to get the food but fail, resulting in a negative reward for travelling. Namely,

$$R(s,a') = \begin{cases} w_1\texttt{Food}(s,a') - w_2\texttt{Distance}(s,a') & \text{Success with probability } p \\ -w_2\texttt{Distance}(s,a') & \text{Failure with probability } 1-p \end{cases}.$$

The challenge of incorporating "attempt" into the model is to determine the value of p. One way to obtain this probability is through calculating the empirical action distributions. We can count the number of times a monkey succeeds or fails to get the food after moving to a specific platform. A more realistic way would be to consider additional aspects that may be relevant to the monkey's chance of getting the food. For example, the success probability p may be lower if moving to a platform that is close to the dominant's current platform.

For "leaving", the immediate reward may be set to a constant that represents the cost of leaving the experiment:

$$R(s, \text{leave}) = \nu,$$

where ν is a non-positive value.

2. Counting how many platforms the monkey has visited and taken the food from.

As mentioned in Section 5.4.2.2, counting the number of visited platforms is particularly relevant for a subordinate monkey, who is likely to stop foraging after visiting a certain number of platforms, even if there is still food left. Following the notations presented in Section 5.4.2.2, we can expand the state representation to include this factor as non-negative integers, namely, $s =$(`f_loc`, `f_food_num`, `remaining_plats`, `c_loc`, `c_food_num`). We can then incorporate the new "attempt" actions into the model, where a successful attempt will increase `f_food_num` by 1 in the next state, and the failed attempt will lead to no increment.

Thus, an example state representation may be $s =$(C1, 2, $(0,0,0,1,1)$, C2, 1), meaning that C1 is the second platform the focal has been to, while C2 is the first platform for the competitor. We may expect that since the competitor has only been to one platform, it is more likely to move to another available platform compared to if `c_food_num` $= 2$. Adding these extra elements (i.e., `f_loc` and `c_loc`) would make the state space more descriptive; however, it would also expand its size. Thus, due to the limited data, it is unclear whether it would potentially lead to better results.

3. Implementing a Markov Game model described in Section 5.4.2.1

As discussed in Section 5.4.2.1, under the assumption that the monkeys are risk-neutral decision makers with bounded intelligence level, we can formulate a Markov game (MG) that is equivalent to our naive level-k model presented in Section 5.2.2.3. However, recall that the objective function for the MG represents the joint likelihood of both agents' unknown parameters and their intelligence level k. Namely,

$$\max_{\hat{p}} \sum_{\tau \in \xi} \log P(\tau|\hat{p}) = \sum_{\tau \in \xi} \log \Pi_{t=0}^{|\tau|-1} P(\hat{a}_t|s_t, \hat{p}), \tag{6.1}$$

where $\hat{p} = (\mathbf{w}^{Dom}, \mathbf{w}^{Sub}, \gamma^{Dom}, \gamma^{Sub})$ represents the parameters to be learned and $P(\hat{a}_t|s_t, \hat{p})$ is

$$P(\hat{a}_t|s_t, \hat{p}) = \pi^{*,i,k^i}(s_t, a_t^i)\pi^{*,-i,k^{-i}}(s_t, a_t^{-i}). \tag{6.2}$$

Therefore, one potential benefit of adapting our model as an MG is that, rather than learning from each monkey's perspective separately, we may obtain a more accurate representation of how the two monkeys forage together by recovering the parameters jointly.

Moreover, since the MG with bounded rationality framework has almost the same structure as the naive level-k model, we believe that it would be a natural extension to our work and a potential starting point for exploring solving the foraging in competition case as a Markov game instead of an MDP.

Extending to Other Foraging Problems

Another potential research direction is to apply our framework to other animal foraging behaviour problems. It would be interesting to see whether the trade-off between food reward and travel distance and the discount factor recovered in our study are relevant in a different foraging context. While finding another experiment that has the same rewards (corn and banana) is difficult, there is one study by Joyce et al. [56] that has a similar setup but with a different platform configuration and food reward (peanuts). The foraging trials were conducted with Japanese macaques and a 6-platform Z-array shown in Figure 6.1.

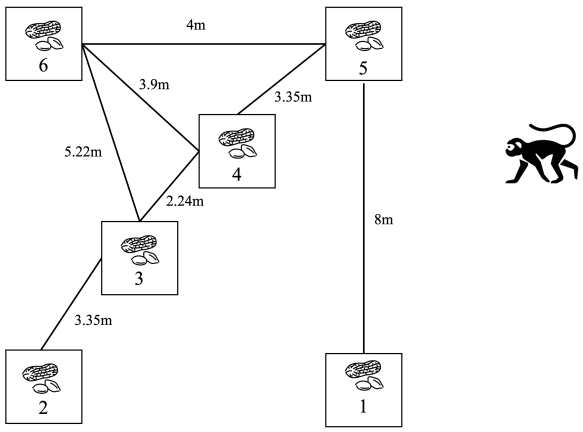


Figure 6.1: A 6-platform Z-array experimental setup in Joyce et al. [56].

Since the experiment is also platform-based, we can easily extend our MDP models to this foraging problem. Moreover, the Z-array setup creates a more diverse environment than the pentagon

array presented in this thesis and there are more strategies a monkey can take to optimize its goals. Each platform in this study was equipped with a camera that captures the monkey's behaviour and records the time taken for foraging accurately, making it ideal for the alternative models we proposed in Chapter 4 and 5.

Extending to Problems in Different Fields

As a future research direction, we can extend our framework to other research areas where the degree to which the decision maker discounts the future affects their choices. One example would be to study human behaviour in the context of an inverse orienteering problem (OP) [43]. We find OP an intuitive extension of the foraging problem as it is a routing problem where we need to decide which subset of nodes (analogous to platforms in our foraging problem) to visit to maximize some notion of reward [45]. For instance, consider a travel itinerary problem where each node is a tourist attraction. Instead of solving the forward problem to determine the best itinerary, if we are instead given a set of trajectories from tourists, can we learn something about their preferences? Applying IRL to such an application can help answer questions such as why tourist A visits attraction A first while tourist B prefers to visit attraction B first. Having a better understanding of tourist behaviour can help attraction sites or travel agencies better plan itineraries to maximize customer satisfaction and potentially use the results from the IRL problem to recommend personalized travel plans.

Another interesting direction related to inverse OP is the Amazon Last Mile challenge,[1] whose goal is to develop solutions for routing problems that outperform traditional optimization techniques. One key aspect of this challenge is to understand how a delivery driver's tacit knowledge impacts their route choices. Similar to the itinerary planning problem mentioned above, we can formulate this problem as an inverse OP where we extract the driver's preferences from delivery trips they have conducted.

Implementing Alternative MaxEnt IRL Algorithms

In this thesis, we extended the original MaxEnt IRL algorithm [129] to solve the foraging problem. For future research, we may try implementing different variations of MaxEnt IRL algorithms and compare their performance against our framework.

One example is the Deep MaxEnt IRL algorithm by Wulfmeier et al. [116], which generalizes MaxEnt IRL to handle non-linear reward functions through a neural network approximation. To adapt Deep MaxEnt IRL to the case with an unknown discount factor, one possible approach is to include the discount factor as an input to the neural network. However, the reward function recovered using this method may not be as interpretable due to the complex structure of a neural network.

Another example is a recent new algorithm proposed by Snoswell et al. [100]. In this study, the authors point out several weaknesses in the original algorithm and present an improved method that leads to better reward learning results. Since the proposed algorithms primarily focus on computing the state visitation frequencies, extending them to the case of learning the discount factor should be trivial since the gradients of the objective function are readily available.

There are many other algorithms that are based on MaxEnt IRL [129] (see Section 2.4.3) and we can explore the possibility of incorporating the unknown discount factor into these methods.

[1] https://routingchallenge.mit.edu/

Dynamic Weight Parameters and Discount Factor

If we have a dataset with recorded timesteps (e.g., data from Joyce et al. [56]), we can not only model the handling time of the banana as discussed in Section 4.5 and 5.4, but also incorporate a time-varying reward and discount factor.

Similar to dynamic IRL proposed by Ashwood et al. [12], we may assume that, during the foraging trial, the monkeys may perceive the reward differently over time. For example, when foraging in competition, a dominant focal may initially prioritize minimizing travel distance, but as the trial progresses and fewer food rewards remain, the focal may shift its focus towards the food itself. That is, the weight parameter for food increases as t increases, while the one for distance decreases.

In addition, we may also introduce a dynamic discount factor, γ_t, that represents how the focal discounts the future over time. In the RL literature, researchers have implemented different adaptive discounting schemes [37, 60] and showed that varying the discount factor during training can improve the final performance of the agent. In our case, the goal of learning γ_t is to recover an interpretable discount factor and weight vector that better explain the elements that affect the animals' decision-making process throughout the experiment.

An Equilibrium-based Approach for the Case of Foraging in Competition

In Section 2.5, we briefly mentioned the connection between multi-agent IRL and inverse game theory. Thus, for the case of foraging in competition, one future direction is to formulate the problem as a game instead of an MDP. Similar to Bertsimas et al. [19], we can treat the given trajectories as observed equilibria of a game between two players: the dominant and the subordinate monkey, and the goal is to recover the players' utility functions, which is equivalent to recovering the reward function in IRL.

There is also existing work in the literature that builds upon MaxEnt IRL for inverse game theory problems [53, 74], making it an intuitive extension for our framework. More specifically, Inga et al. [53] present solution methods for several types of games, including non-cooperative games, where the players are in competition with each other. The gradient-based approach proposed in Inga et al. [53] is similar to our method presented in Section 3.3 and thus we should be able to extend to the case of learning the discount factor.

Formulating the problem from a game-theoretic perspective not only allows us to better study each player's strategy but also offers an opportunity to further establish the connection between IRL and inverse game theory.

6.3 Concluding Remarks

The goal of this thesis is to explore the framework of IRL with unknown discount factor and apply it to an application in animal behaviour. This work is built upon the method proposed by Giwa and Lee [40, 41] and we studied the foraging behaviour of wild vervet monkeys as an IRL problem. We investigated two specific foraging scenarios: alone and in competition, and demonstrated the monkeys value the future rewards differently depending on if a competitor is present and on their social status. We hope this work can serve as a starting point for future research on animal behaviour using IRL.

Bibliography

[1] Nematollah Ab Azar, Aref Shahmansoorian, and Mohsen Davoudi. "From inverse optimal control to inverse reinforcement learning: A historical review". In: *Annual Reviews in Control* 50 (2020), pp. 119–138.

[2] Pieter Abbeel, Adam Coates, Timothy Hunter, and Andrew Y Ng. "Autonomous autorotation of an RC helicopter". In: *Experimental robotics*. Springer. 2009, pp. 385–394.

[3] Pieter Abbeel, Adam Coates, and Andrew Y Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[4] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Ng. "An application of reinforcement learning to aerobatic helicopter flight". In: *Advances in neural information processing systems* 19 (2006).

[5] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[6] Navid Aghasadeghi and Timothy Bretl. "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 1561–1566.

[7] Paul CH Albers and Han de Vries. "Elo-rating as a tool in the sequential estimation of dominance strengths". In: *Animal Behaviour* (2001), pp. 489–495.

[8] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[9] Saurabh Arora and Prashant Doshi. "A survey of inverse reinforcement learning: Challenges, methods and progress". In: *Artificial Intelligence* 297 (2021), p. 103500.

[10] T Jean M Arseneau-Robar, Karyn A Anderson, Eric N Vasey, Pascale Sicotte, and Julie A Teichroeb. "Think Fast!: Vervet Monkeys Assess the Risk of Being Displaced by a Dominant Competitor When Making Foraging Decisions". In: *Frontiers in Ecology and Evolution* 10 (2022), p. 775288.

[11] Kavosh Asadi and Michael L Littman. "An alternative softmax operator for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 243–252.

[12] Zoe Ashwood, Aditi Jha, and Jonathan W Pillow. "Dynamic Inverse Reinforcement Learning for Characterizing Animal Behavior". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 29663–29676.

[13] Zoe Ashwood, Aditi Jha, and Jonathan W Pillow. "Dynamic Inverse Reinforcement Learning for Characterizing Animal Behavior". In: *Advances in Neural Information Processing Systems*.

[14] Julien Audiffren, Michal Valko, Alessandro Lazaric, and Mohammad Ghavamzadeh. "Maximum entropy semi-supervised inverse reinforcement learning". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[15] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. "Bayesian theory of mind: Modeling joint belief-desire attribution". In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 33. 33. 2011.

[16] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. "Action understanding as inverse planning". In: *Cognition* 113.3 (2009), pp. 329–349.

[17] Chris L Baker and Joshua B Tenenbaum. "Modeling human plan recognition using Bayesian theory of mind". In: *Plan, activity, and intent recognition: Theory and practice* 7 (2014), pp. 177–204.

[18] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[19] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. "Data-driven estimation in equilibrium using inverse optimization". In: *Mathematical Programming* 153 (2015), pp. 595–633.

[20] Michael Bloem and Nicholas Bambos. "Infinite time horizon maximum causal entropy inverse reinforcement learning". In: *53rd IEEE conference on decision and control*. IEEE. 2014, pp. 4911–4916.

[21] Kenneth Bogert and Prashant Doshi. "Multi-robot inverse reinforcement learning under occlusion with estimation of state transitions". In: *Artificial Intelligence* 263 (2018), pp. 46–73.

[22] Abdeslam Boularias, Jens Kober, and Jan Peters. "Relative entropy inverse reinforcement learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 182–189.

[23] Daniel S Brown and Scott Niekum. "Deep bayesian reward learning from preferences". In: *arXiv preprint arXiv:1912.04472* (2019).

[24] Colin F Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton university press, 2011.

[25] Kun Cao and Lihua Xie. "Game-Theoretic Inverse Reinforcement Learning: A Differential Pontryagin's Maximum Principle Approach". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[26] Jaedeug Choi and Kee-Eung Kim. "Hierarchical bayesian inverse reinforcement learning". In: *IEEE transactions on cybernetics* 45.4 (2014), pp. 793–805.

[27] Jaedeug Choi and Kee-Eung Kim. "Map inference for bayesian inverse reinforcement learning". In: *Advances in Neural Information Processing Systems* 24 (2011).

[28] Jaedeug Choi and Kee-Eung Kim. "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions". In: *Advances in neural information processing systems* 25 (2012).

[29] Adam Coates, Pieter Abbeel, and Andrew Y Ng. "Learning for control from multiple demonstrations". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 144–151.

[30] M Costa-Gomes, VP Crawford, and B Broseta. "Cognition and Behavior in Normal-Form Games: An Experimental Study', Econometrica, 69 (5), September, 1193-235". In: *INTERNATIONAL LIBRARY OF CRITICAL WRITINGS IN ECONOMICS* 209.1 (2007), p. 279.

[31] Sanmay Das and Allen Lavoie. "The effects of feedback on human behavior in social media: An inverse reinforcement learning model". In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. Citeseer. 2014, pp. 653–660.

[32] Daniel C Dennett. *The intentional stance*. MIT press, 1989.

[33] Arpad E Elo and Sam Sloan. "The rating of chessplayers: Past and present". In: *(No Title)* (1978).

[34] Muhammad Fahad, Zhuo Chen, and Yi Guo. "Learning how pedestrians navigate: A deep inverse reinforcement learning approach". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 819–826.

[35] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models". In: *arXiv preprint arXiv:1611.03852* (2016).

[36] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". In: *International conference on machine learning*. PMLR. 2016, pp. 49–58.

[37] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. "How to discount deep reinforcement learning: Towards new dynamic strategies". In: *arXiv preprint arXiv:1512.02011* (2015).

[38] Justin Fu, Katie Luo, and Sergey Levine. "Learning robust rewards with adversarial inverse reinforcement learning". In: *arXiv preprint arXiv:1710.11248* (2017).

[39] Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi, and Giacomo Rizzolatti. "Action recognition in the premotor cortex". In: *Brain* 119.2 (1996), pp. 593–609.

[40] Babatunde H Giwa and Chi-Guhn Lee. "Estimation of Discount Factor in a Model-Based Inverse Reinforcement Learning Framework". In: *Bridging the Gap Between AI Planning and Reinforcement Learning Workshop at ICAPS*. 2021.

[41] Babatunde Halar Giwa. "Discount Factor Estimation in Inverse Reinforcement Learning". PhD thesis. University of Toronto (Canada), 2022.

[42] Adam Gleave and Sam Toyer. "A Primer on Maximum Causal Entropy Inverse Reinforcement Learning". In: *arXiv preprint arXiv:2203.11409* (2022).

[43] Bruce L Golden, Larry Levy, and Rakesh Vohra. "The orienteering problem". In: *Naval Research Logistics (NRL)* 34.3 (1987), pp. 307–318.

[44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[45] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. "Orienteering problem: A survey of recent variants, solution approaches and applications". In: *European Journal of Operational Research* 255.2 (2016), pp. 315–332.

[46] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. "Inverse reward design". In: *Advances in neural information processing systems* 30 (2017).

[47] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. "Cooperative inverse reinforcement learning". In: *Advances in neural information processing systems* 29 (2016).

[48] Cesar Hernandez-Reyes, Shunsuke Shigaki, Mayu Yamada, Takeshi Kondo, and Daisuke Kurabayashi. "Learning a Generic Olfactory Search Strategy From Silk Moths by Deep Inverse Reinforcement Learning". In: *IEEE Transactions on Medical Robotics and Bionics* 4.1 (2021), pp. 241–253.

[49] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, Hironobu Fujiyoshi, Yuta Umezu, Ichiro Takeuchi, Sakiko Matsumoto, and Ken Yoda. "Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning". In: *Ecosphere* 9.10 (2018), e02447.

[50] Ronald A Howard. "Dynamic programming and markov processes." In: (1960).

[51] Junling Hu and Michael P Wellman. "Nash Q-learning for general-sum stochastic games". In: *Journal of machine learning research* 4.Nov (2003), pp. 1039–1069.

[52] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[53] Jairo Inga, Esther Bischoff, Florian Köpf, and Sören Hohmann. "Inverse dynamic games based on maximum entropy inverse reinforcement learning". In: *arXiv preprint arXiv:1911.07503* (2019).

[54] Julian Jara-Ettinger. "Theory of mind as inverse reinforcement learning". In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 105–110.

[55] Edwin T Jaynes. "Information theory and statistical mechanics". In: *Physical review* 106.4 (1957), p. 620.

[56] Megan M Joyce, Julie A Teichroeb, Yu Kaigaishi, Brogan M Stewart, Kazunori Yamada, and Sarah E Turner. "No food left behind: foraging route choices among free-ranging Japanese macaques (Macaca fuscata) in a multi-destination array at the Awajishima Monkey Center, Japan". In: *Primates* (2023), pp. 1–17.

[57] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[58] Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. "Learning objective functions for manipulation". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 1331–1336.

[59] Parameswaran Kamalaruban, Rati Devidze, Volkan Cevher, and Adish Singla. "Interactive teaching algorithms for inverse reinforcement learning". In: *arXiv preprint arXiv:1905.11867* (2019).

[60] MyeongSeop Kim, Jung-Su Kim, Myoung-Su Choi, and Jae-Han Park. "Adaptive Discount Factor for Deep Reinforcement Learning in Continuing Tasks with Uncertainty". In: *Sensors* 22.19 (2022), p. 7266.

[61] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. "Inverse reinforcement learning through structured classification". In: *Advances in neural information processing systems* 25 (2012).

[62] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.

[63] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. "Learning driving styles for autonomous vehicles from demonstration". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2641–2646.

[64] Volodymyr Kuleshov and Okke Schrijvers. "Inverse game theory: Learning utilities in succinct games". In: *Web and Internet Economics: 11th International Conference, WINE 2015, Amsterdam, The Netherlands, December 9-12, 2015, Proceedings 11*. Springer. 2015, pp. 413–427.

[65] Sergey Levine, Zoran Popovic, and Vladlen Koltun. "Feature construction for inverse reinforcement learning". In: *Advances in neural information processing systems* 23 (2010).

[66] Sergey Levine, Zoran Popovic, and Vladlen Koltun. "Nonlinear inverse reinforcement learning with gaussian processes". In: *Advances in neural information processing systems* 24 (2011).

[67] Xiaomin Lin, Stephen C Adams, and Peter A Beling. "Multi-agent inverse reinforcement learning for general-sum stochastic games". In: *arXiv preprint arXiv:1806.09795* (2018).

[68] Xiaomin Lin, Peter A Beling, and Randy Cogill. "Multiagent inverse reinforcement learning for two-person zero-sum games". In: *IEEE Transactions on Games* 10.1 (2017), pp. 56–68.

[69] Michael L Littman. "Markov games as a framework for multi-agent reinforcement learning". In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[70] Jundi Liu, Linda Ng Boyle, and Ashis G Banerjee. "An inverse reinforcement learning approach for customizing automated lane change systems". In: *IEEE Transactions on Vehicular Technology* 71.9 (2022), pp. 9261–9271.

[71] Siyuan Liu, Miguel Araujo, Emma Brunskill, Rosaldo Rossetti, Joao Barros, and Ramayya Krishnan. "Understanding sequential decisions via inverse reinforcement learning". In: *2013 IEEE 14th International Conference on Mobile Data Management*. Vol. 1. IEEE. 2013, pp. 177–186.

[72] Manuel Lopes, Francisco Melo, and Luis Montesano. "Active learning for reward estimation in inverse reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, pp. 31–46.

[73] Tien Mai, Kennard Chan, and Patrick Jaillet. "Generalized maximum causal entropy for inverse reinforcement learning". In: *arXiv preprint arXiv:1911.06928* (2019).

[74] Negar Mehr, Mingyu Wang, Maulik Bhatt, and Mac Schwager. "Maximum-entropy multi-agent dynamic games: Forward and inverse solutions". In: *IEEE Transactions on Robotics* (2023).

[75] Bernard Michini and Jonathan P How. "Bayesian nonparametric inverse reinforcement learning". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2012, pp. 148–163.

[76] Bernard Michini and Jonathan P How. "Improving the efficiency of Bayesian inverse reinforcement learning". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3651–3656.

[77] Walter Mischel and Ebbe B Ebbesen. "Attention in delay of gratification." In: *Journal of personality and social psychology* 16.2 (1970), p. 329.

[78] Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf, and Jan Peters. "Learning strategies in table tennis using inverse reinforcement learning". In: *Biological cybernetics* 108.5 (2014), pp. 603–619.

[79] Rosemarie Nagel. "Unraveling in guessing games: An experimental study". In: *The American economic review* 85.5 (1995), pp. 1313–1326.

[80] Sriraam Natarajan, Gautam Kunapuli, Kshitij Judah, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. "Multi-agent inverse reinforcement learning". In: *2010 ninth international conference on machine learning and applications*. IEEE. 2010, pp. 395–400.

[81] Gergely Neu and Csaba Szepesvári. "Apprenticeship learning using inverse reinforcement learning and gradient methods". In: *arXiv preprint arXiv:1206.5264* (2012).

[82] Andrew Y Ng, Stuart Russell, et al. "Algorithms for inverse reinforcement learning." In: *ICML*. Vol. 1. 2000, p. 2.

[83] Billy Okal and Kai O Arras. "Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 2889–2895.

[84] Max Pflueger, Ali Agha, and Gaurav S Sukhatme. "Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1387–1394.

[85] Robert Pinsler, Max Maag, Oleg Arenz, and Gerhard Neumann. "Inverse reinforcement learning of bird flocking behavior". In: *ICRA Swarms Workshop*. 2018.

[86] Qifeng Qiao and Peter A Beling. "Inverse reinforcement learning with Gaussian process". In: *Proceedings of the 2011 American control conference*. IEEE. 2011, pp. 113–118.

[87] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning." In: *IJCAI*. Vol. 7. 2007, pp. 2586–2591.

[88] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. "Maximum margin planning". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 729–736.

[89] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely Zaruba, and Manfred Huber. "Inverse reinforcement learning for decentralized non-cooperative multiagent systems". In: *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2012, pp. 1930–1935.

[90] Nicholas Rhinehart and Kris M Kitani. "First-person activity forecasting with online inverse reinforcement learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3696–3705.

[91] Matthew Rosenberg, Tony Zhang, Pietro Perona, and Markus Meister. "Mice in a labyrinth show rapid learning, sudden insight, and efficient exploration". In: *Elife* 10 (2021), e66175.

[92] Stuart Russell. "Learning agents for uncertain environments". In: *Proceedings of the eleventh annual conference on Computational learning theory*. 1998, pp. 101–103.

[93] Stefan Schaal. "Learning from demonstration". In: *Advances in neural information processing systems* 9 (1996).

[94] Toryn LJ Schafer, Christopher K Wikle, and Mevin B Hooten. "Bayesian inverse reinforcement learning for collective animal movement". In: *The Annals of Applied Statistics* 16.2 (2022), pp. 999–1013.

[95] Robert M Seyfarth and Dorothy L Cheney. "Grooming, alliances and reciprocal altruism in vervet monkeys". In: *Nature* 308.5959 (1984), pp. 541–543.

[96] Claude E Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[97] Sahand Sharifzadeh, Ioannis Chiotellis, Rudolph Triebel, and Daniel Cremers. "Learning to drive using inverse reinforcement learning and deep q-networks". In: *arXiv preprint arXiv:1612.03653* (2016).

[98] Kyriacos Shiarlis, Joao Messias, and SA Whiteson. "Inverse reinforcement learning from failure". In: (2016).

[99] Herbert A Simon. "From substantive to procedural rationality". In: *25 years of economic theory: Retrospect and prospect* (1976), pp. 65–86.

[100] Aaron J Snoswell, Surya PN Singh, and Nan Ye. "Revisiting maximum entropy inverse reinforcement learning: new perspectives and algorithms". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2020, pp. 241–249.

[101] Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. "Inverse reinforcement learning in swarm systems". In: *arXiv preprint arXiv:1602.05450* (2016).

[102] Dale O Stahl and Paul W Wilson. "On players' models of other players: Theory and experimental evidence". In: *Games and Economic Behavior* 10.1 (1995), pp. 218–254.

[103] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[104] Umar Syed and Robert E Schapire. "A game-theoretic approach to apprenticeship learning". In: *Advances in neural information processing systems* 20 (2007).

[105] Bulent Tastan and Gita Sukthankar. "Learning policies for first person shooter games using inverse reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 7. 1. 2011, pp. 85–90.

[106] Ran Tian, Liting Sun, and Masayoshi Tomizuka. "Bounded risk-sensitive markov games: Forward policy design and inverse reward learning with iterative reasoning and cumulative prospect theory". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 7. 2021, pp. 6011–6020.

[107] Niko Tinbergen. "On aims and methods of ethology". In: *Zeitschrift für tierpsychologie* 20.4 (1963), pp. 410–433.

[108] Eiji Uchibe. "Model-free deep inverse reinforcement learning by logistic regression". In: *Neural Processing Letters* 47.3 (2018), pp. 891–905.

[109] Michal Valko, Mohammad Ghavamzadeh, and Alessandro Lazaric. "Semi-supervised apprenticeship learning". In: *European workshop on reinforcement learning*. PMLR. 2013, pp. 131–142.

[110] Dizan Vasquez, Billy Okal, and Kai O Arras. "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1341–1346.

[111] Baoxiang Wang, Tongfang Sun, and Xianjun Sam Zheng. "Beyond winning and losing: modeling human motivations and behaviors using inverse reinforcement learning". In: *arXiv preprint arXiv:1807.00366* (2018).

[112] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. "Computational rationalization: The inverse equilibrium problem". In: *arXiv preprint arXiv:1308.3506* (2013).

[113] Stephen J Wright. "Coordinate descent algorithms". In: *Mathematical programming* 151.1 (2015), pp. 3–34.

[114] Zheng Wu, Fangbing Qu, Lin Yang, and Jianwei Gong. "Human-like decision making for autonomous vehicles at the intersection using inverse reinforcement learning". In: *Sensors* 22.12 (2022), p. 4500.

[115] Zheng Wu, Liting Sun, Wei Zhan, Chenyu Yang, and Masayoshi Tomizuka. "Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5355–5362.

[116] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. "Maximum entropy deep inverse reinforcement learning". In: *arXiv preprint arXiv:1507.04888* (2015).

[117] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. "Large-scale cost function learning for path planning using deep inverse reinforcement learning". In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1073–1087.

[118] Shoichiro Yamaguchi, Honda Naoki, Muneki Ikeda, Yuki Tsukada, Shunji Nakano, Ikue Mori, and Shin Ishii. "Identification of animal behavioral strategies by inverse reinforcement learning". In: *PLoS computational biology* 14.5 (2018), e1006122.

[119] Yaodong Yang and Jun Wang. "An overview of multi-agent reinforcement learning from game theoretical perspective". In: *arXiv preprint arXiv:2011.00583* (2020).

[120] Zhibo Yang, Lihan Huang, Yupei Chen, Zijun Wei, Seoyoung Ahn, Gregory Zelinsky, Dimitris Samaras, and Minh Hoai. "Predicting goal-directed human attention using inverse reinforcement learning". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 193–202.

[121] Lantao Yu, Jiaming Song, and Stefano Ermon. "Multi-agent adversarial inverse reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7194–7201.

[122] Xin Yu, Wenjun Wu, Pu Feng, and Yongkai Tian. "Swarm inverse reinforcement learning for biological systems". In: *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2021, pp. 274–279.

[123] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. "Xirl: Cross-embodiment inverse reinforcement learning". In: *Conference on Robot Learning*. PMLR. 2022, pp. 537–546.

[124] Xiangyuan Zhang, Kaiqing Zhang, Erik Miehling, and Tamer Basar. "Non-cooperative inverse reinforcement learning". In: *Advances in neural information processing systems* 32 (2019).

[125] Zhouqiao Zhao, Ziran Wang, Kyungtae Han, Rohit Gupta, Prashant Tiwari, Guoyuan Wu, and Matthew J Barth. "Personalized car following for autonomous driving with inverse reinforcement learning". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2891–2897.

[126] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. "Robust bayesian inverse reinforcement learning with sparse behavior noise". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[127] J Zico Kolter and Andrew Y Ng. "The stanford littledog: A learning and rapid replanning approach to quadruped locomotion". In: *The International Journal of Robotics Research* 30.2 (2011), pp. 150–174.

[128] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. "Modeling interaction via the principle of maximum causal entropy". In: *ICML*. 2010.

[129] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[130] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. "Planning-based prediction for pedestrians". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 3931–3936.